Markt&Technik

Olaf Hartwig

EXPERIMENTE ZUR KÜNSTLICHEN INTELLIGENZ in BASIC auf CPC 464/664/6128

Eine praxisbezogene Einführung
in das Verarbeiten natürlicher Sprache,
Wissensrepräsentation, Computer-Kreativität,
Robotics und Expertensysteme.

Experimente zur künstlichen Intelligenz
in BASIC auf CPC 464/664/6128

Olaf Hartwig

Experimente zur künstlichen Intelligenz in BASIC auf CPC 464/664/6128

Eine praxisbezogene Einführung in das Verarbeiten natürlicher Sprache, Wissensrepräsentation, Computer-Kreativität, Robotics und Expertensysteme.

Markt&Technik Verlag AG

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Hartwig, Olaf:
Experimente zur künstlichen Intelligenz in BASIC auf CPC 464, 664, 6128 :
e. praxisbezogene Einf. in d. Verarbeiten natürl. Sprache, Wissensrepräsentation,
Computer-Kreativität, Robotics u. Expertensysteme / Olaf Hartwig. –
Haar bei München : Markt-und-Technik-Verlag, 1987.
ISBN 3-89090-473-4

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
90 89 88 87

ISBN 3-89090-473-4

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Jantsch, Günzburg
Printed in Germany

Inhaltsverzeichnis

Einleitung

Künstliche Intelligenz, kurz KI genannt, ist sicher eines der faszinierendsten, gleichzeitig aber auch ein sehr umstrittenes Feld der modernen Computer-Forschung. Sind Maschinen intelligent? Können Computer denken? Wenn ja, können sie auch kreativ sein, sind sie in der Lage, so etwas wie Fantasie zu entwickeln? Was eigentlich ist Künstliche Intelligenz? Hier erhalten Sie die Antworten auf diese Fragen.
Das vorliegende Buch gliedert sich in zwei große Teile, in einen Praxisteil und einen Teil, der sich gezielt mit Hintergrundwissen zur Künstlichen Intelligenz befaßt.
Im ersten Teil können Sie anhand vielfältiger KI-Programme praktisch die Möglichkeiten, aber auch die Grenzen der KI erfahren. Die ausführlich dokumentierten Programme, die speziell für den CPC entwickelt wurden, eröffnen interessante Einblicke in das Wesen der Künstlichen Intelligenz und ermuntern zu eigenen Experimenten:

»Bringen Sie Ihren Computer dazu, natürliche Sprache, also normales Umgangsdeutsch, zu verstehen und intelligente Antworten zu geben. Dazu werden Ihnen alle entsprechenden Techniken der Künstlichen Intelligenz zusammen mit Realisierungen in verständlichen Programmen vorgestellt.
Anschließend befassen wir uns mit dem Thema der Computer-Kreativität. Im Mittelpunkt der Betrachtungen steht dabei die automatische Generierung von Gedichten und kompletten Texten. Außerdem wird ein Seitenblick auf Computer-Kunst und Kreativität geworfen.
Der in letzter Zeit immer wichtiger werdende Bereich von Künstlicher Intelligenz und Robotics bildet einen wesentlichen Schwerpunkt dieses Buches. Am Beispiel des Micro-Mouse-Wettbewerbs werden Ihnen die Grundlagen der Robotics sowie eine Reihe intelligenter Micro-Mouse-Simulationsprogramme vorgestellt. Ausgehend von diesen praktischen Erfahrungen wird Ihnen sowohl interessantes Hintergrundwissen über intelligente Roboter der fünften Generation als auch über aktuelle Robotics-Trends wie den Roboter Ping-Pong-Wettbewerb u.v.a. vermittelt.
Heute beginnt die Künstliche Intelligenz vor allem durch Expertensysteme in die Arbeitswelt vorzudringen. Anhand vieler Beispiele wird demonstriert, was ein Expertensystem ist, wie es strukturiert ist, was es zu leisten vermag und wo seine speziellen Einsatzbereiche liegen.«

Der erste Abschnitt des Buches besteht aus einer Darstellung verschiedener Gebiete der KI-Forschung. Die zu jedem Gebiet notwendigen Programmiertechniken werden in leicht nachvollziehbaren Schritten eingeführt.
Der zweite Teil stellt ein umfassendes KI-Forschungsprofil dar. Es reicht von einer Darstellung aller KI-Disziplinen, deren Methoden, über eine KI-Chronologie bis hin zu den Zielen und Anwendungsbereichen der KI. Daran schließt sich ein Blick in die Zukunft der Künstlichen Intelligenz und ihre Auswirkungen an.
Ich wünsche Ihnen viel Freude bei der Ergründung der Möglichkeiten der Künstlichen Intelligenz und Robotics mit dem CPC.

Olaf Hartwig

1 Stellenwert der Künstlichen Intelligenz

Bis Mitte der siebziger Jahre wurde die Forschung auf dem Gebiet der Künstlichen Intelligenz noch als »Spielerei« einiger Wissenschaftler angesehen. Doch in den achtziger Jahren begann sich diese Einschätzung zu ändern. Zunächst wurde das Militär auf die vielfältigen Möglichkeiten der KI-Techniken aufmerksam, später gesellten sich immer mehr zivile Anwender hinzu, die sich vor allem den Experten-Systemen, sogenannten intelligenten Assistenten, und dem Bereich der Robotics widmeten.

Wie ist dieser Umschwung zu erklären? Die Künstliche Intelligenz existiert als Forschungsbereich seit über 25 Jahren. Warum aber wurden beispielsweise die meisten Publikationen zu diesem Thema erst in den letzten fünf Jahren veröffentlicht?

Viele der neu entstandenen KI-Anwendungen basieren auf Forschungsergebnissen, die vor teilweise 10 bis 20 Jahren erzielt wurden. Die Ergebnisse der frühen KI-Forschung bilden die wesentliche Grundlage der heutigen anwendungsorientierten KI-Systeme. Im wesentlichen ist die sehr späte Erkenntnis der Möglichkeiten der Künstlichen Intelligenz darauf zurückzuführen, daß die heutigen Anwender in vielen Fällen erst recht spät die Existenz des Wissenschaftszweiges KI zur Kenntnis nahmen. Vor allem die meisten praktischen Anwendungsmöglichkeiten wurden erst Ende der siebziger Jahre bemerkt.

Zu diesem Zeitpunkt hat die Wirtschaft erkannt, daß sich mit der Anwendung und Vermarktung der Künstlichen Intelligenz ein finanzieller Gewinn erzielen läßt. Vor allem aus diesem Grund wurde in den Forschungszweig der KI kräftig investiert.

Die direkten Folgen sind vielen Menschen auch heute noch gar nicht so recht bewußt. Beispiele für die immer stärker zunehmende Verbreitung Künstlicher Intelligenz sind nukleare, interkontinentale Marschflugkörper, die ihren Weg ins Ziel mittels KI finden.

Auch intelligente Roboter der fünften Generation finden immer häufiger ihren Einsatz in den Fabriken der Industrienationen. Die mit auf Software basierender Intelligenz ausgestatteten Roboter können sehen, fühlen und selbständig Pläne entwickeln. So haben sie u.a. die Fähigkeit, sich an unvorhergesehene Umstände oder an verschiedene Umweltbedingungen anzupassen. Sie können bei Bedarf auch eine gesamte Produktionspalette in kurzer Zeit umstellen.

Noch größere Folgen zeitigt die KI auf dem Gebiet der Expertensysteme. Die intelligenten Assi-

stenten, die mit einem großen Maß an Wissen auf ihrem Spezialgebiet ausgestattet sind, haben ihren Einzug in Industrie und Forschungslabore rund um die Welt bereits vor einigen Jahren gehalten. Sie sind aus den Bereichen der Medizin, der Meteorologie, der Luft- und Raumfahrt, der Chemie und unzähligen anderen nicht mehr wegzudenken. Viele Computer-Firmen wie IBM, Nixdorf, Sperry, TI, Xerox, Siemens und zahlreiche japanische EDV-Firmen entwickeln an bzw. forschen mit derartigen Systemen.

Eine auslösende Wirkung für die intensive KI-Forschung erzielten die Japaner mit ihrem 1981 ins Leben gerufenen Projekt der Entwicklung von Computern der fünften Generation. Innerhalb der Ende 1984 abgeschlossenen ersten Phase des auf insgesamt 10 Jahre angelegten Programms konnten bereits bemerkenswerte Erfolge erzielt werden.

Leider ist es auch so, daß ein entscheidender Motor der KI-Forschung das Militär ist. So werden z.B. die amerikanischen Cruise Missiles durch das aus der KI-Entwicklung stammende »Tercom Guidance System« gelenkt. Dieses intelligente System vergleicht die Landschaft unter sich mit einer gespeicherten Landkarte und paßt den Kurs der Rakete den Bodengegebenheiten an bzw. führt alle nötigen Kurskorrekturen selbständig durch.

Schon seit längerem setzen die Militärstrategen im Pentagon auch Expertensysteme ein, die beispielsweise See- oder Luftkriege mit jeweils unterschiedlichen Parametern durchspielen.

Was ist Künstliche Intelligenz?

An den vorangegangenen Ausführungen ist Ihnen sicher deutlich geworden, welchen Stellenwert die Künstliche Intelligenz einnimmt. Nachdem Sie die KI-Programme dieses Buches ausprobiert haben, werden Sie die Möglichkeiten, aber auch die Grenzen der KI genauer einschätzen können.

Zuvor ist es jedoch sinnvoll, sich eine Definition für Künstliche Intelligenz zu überlegen. Leider gibt es hier fast genauso viele Ansätze wie auf diesem Gebiet forschende Wissenschaftler. Das liegt vor allem daran, daß es auch heute immer noch keine allgemeingültige Definition für normale Intelligenz gibt.

Was ist Intelligenz? Unter Psychologen geht der Witz um, Intelligenz sei eben das, was sich durch Intelligenztests messen ließe. Auf der Weltkonferenz der KI, der International Joint Conference on Artificial Intelligence definierte einer der Organisatoren, Dr. Jörg Siekmann, die Künstliche Intelligenz als den Versuch, intellektuelle Fähigkeiten, die bisher den Menschen vorbehalten waren, auf einem Computer zu realisieren. Als Beispiele führte er einige Spezialgebiete der KI an:

»In Experten-Systemen etwa versucht man, den Computer als Experten für bestimmte Spezialgebiete auszubilden, also zum Beispiel als medizinischen Experten, als Experten für die Ölsuche und so weiter. Die Programme, die das machen, leisten schon heute mehr als die meisten Fachleute auf den entsprechenden Gebieten.

Andere Bereiche sind die Gestaltwahrnehmung, die Sprachverarbeitung und schließlich Robotics – also die Steuerung von Handhabungsautomaten oder Robotern –, deren Einsatzmöglichkeiten sich bei der Hinzunahme kognitiver Fähigkeiten drastisch erweitern lassen.«

Es stellt sich dabei allerdings die Frage, ob diese Anwendungen nun wirklich intelligent sind, oder ob es sich dabei nur um geschickt konzipierte Programme handelt. Einer der Väter der Computer-Technik, Alan M. Turing, ist folgender Auffassung:

»Alles, was man mit Symbolen machen kann, läßt sich auch mit einem ganz gewöhnlichen Computer machen.«

Der Computer-Wissenschaftler und Nobelpreisträger Herbert A. Simon (Carnegie Mellon University) nimmt diesen Grundgedanken in seiner bekannten These über Symbolsysteme auf:

»Ein System, das die Fähigkeit besitzt, Symbole zu lesen, schreiben, vergleichen, löschen, erzeugen und zu übertragen, erfüllt die notwendige und hinreichende Bedingung, das zu tun, was wir denken nennen.«

Nach diesem Modell sind Computer sicher in der Lage, zu denken. Ob man das Modell jedoch persönlich akzeptiert, hängt von den individuellen Akzenten ab, die man setzt. Viele Menschen sind beispielsweise der Auffassung, daß Intelligenz nur mit zugehörigem Lernvermögen wirkliche Intelligenz ist.
Allerdings besitzen einige moderne KI-Systeme bereits diese Fähigkeit. Sie können also von selber lernen und Wissen bzw. Regeln überprüfen und auch als falsch erkennen.
Skeptiker bringen noch das Argument der Kreativität in die Diskussion über Computer-Intelligenz ein. Können Computer kreativ sein? Können sie aus sich selbst mit Intuition und Inspiration Neues erschaffen?
Die Frage ist unter KI-Experten umstritten. Genauso unklar ist unter Fachleuten die Frage, ob Kreativität zu Intelligenz notwendig ist. Um die Problematik einmal kurz anzudeuten: Was unterscheidet zum Beispiel eine Komposition von Beethoven von einer einfachen Schlagerkomposition? Wo liegen die Unterschiede zwischen einem Trivialroman und einer niveauvollen Lektüre? Erfordert das Erstellen von Trivialromanen Kreativität? Wenn es so sein sollte, so könnte man auch Computer als kreativ einstufen: Kürzlich wurde ein KI-System entwickelt, das selbständig Drehbücher zu den TV-Serien »Denver« und »Dallas« schreibt.
Eine andere Überlegung zur Computer-Intelligenz: Die NASA plant beispielsweise, ihren eindrucksvollen Kontrollraum in Houston durch ein einziges Expertensystem zu ersetzen. Die geplante permanente Raumstation erfordere dies, erklärte kürzlich der NASA-Experte Jerry Mazlack auf der letzten Weltkonferenz der KI. Sind die hochbezahlten menschlichen Experten, die dann von einem Computer-System ersetzt werden, nicht intelligent?
Die in diesem Abschnitt vorgestellten Vorüberlegungen und Beispiele mögen die Problematik des Forschungsbereichs der KI ansatzweise skizziert haben. Beenden wir damit vorerst die theoretischen Überlegungen zur Künstlichen Intelligenz und wenden uns nun praktischen und für Sie direkt am Computer nachvollziehbaren Experimenten, den Grundlagen der Künstlichen Intelligenz und der Robotics zu.

2 Verarbeiten natürlicher Sprache

2.1 Ansätze zur Sprachverarbeitung

Schon seit Mitte der fünfziger Jahre versucht die KI-Forschung, Computern beizubringen, wie sie natürlichsprachige Sätze erzeugen bzw. verstehen können. Auf den ersten Blick scheint das Problem nicht allzu schwierig zu sein. Schließlich versteht auch der Mensch nur ein begrenztes Vokabular, welches durch eine ebenfalls begrenzte Anzahl grammatischer Regeln koordiniert wird.

Doch die Schwierigkeiten stecken hier im Detail. Tatsächlich ist dieser Forschungszweig der KI einer der problematischsten und Fortschritte stellen sich nur sehr zögernd ein. Das Hauptproblem liegt hier in der Unschärfe der natürlichen Sprache. Für einen Computer ist es momentan noch fast unmöglich, doppeldeutige Wörter richtig zuzuordnen. Auch sind die grammatischen Regeln nicht konkret wie z.B. mathematische Regeln, sondern vielfältig auslegbar und anwendbar.

Einige der Probleme der Verarbeitung natürlicher Sprache werden Ihnen noch deutlicher, wenn wir zusammen einige Programme zur Verarbeitung natürlicher Sprache entwickeln. Zuvor wollen wir erst einmal klären, welches die Hauptprinzipien bei der Entwicklung derartiger Programme sind.

Die Sprachverarbeitung durch den Computer zielt in zwei grundsätzliche Richtungen. Zum einen sollen automatische Sprachübersetzer entwickelt und verbessert werden. Die Fehlerquote derartiger Software-Pakete liegt momentan noch bei fast 20 Prozent. Dieser Prozentsatz muß noch vom Menschen manuell nachübersetzt werden.

Ein Interessent dieser Entwicklung war hier das Militär. Seit über 20 Jahren wird beispielsweise fieberhaft an einem Übersetzer vom Russischen in das Amerikanische gearbeitet.

Hauptziel der maschinellen Sprachverarbeitung ist eine Mensch-Maschine-Schnittstelle in natürlicher Sprache. Diese soll es dem Benutzer ermöglichen, mit einem Computer in seiner Muttersprache zu kommunizieren. Dadurch soll es beispielsweise möglich sein, aus Datenbanksystemen auf einfachste Weise die jeweils benötigten Informationen zu erhalten.

Um die zu erreichen, muß die Software fähig sein, die über die Tastatur eingegebene Sprache syntaktisch analysieren zu können. Gleichzeitig muß sie sie auch verstehen, also den Sprachinhalt intern für sich aufbereiten, d.h., sie muß die Sprache intern repräsentieren können. Mit Hilfe einer Wissensbasis, einer Datenbank, muß sie dann in einem grammatikalisch richtigen Format sinnvolle, d.h. auf die Eingaben des Benutzers bezogene Antworten generieren.
Der Themenkomplex, also die syntaktische Analyse der vom Menschen eingegebenen natürlichen deutschen Sprache sowie die Repräsentation dieser Sprache in einem dem Computer verständlichen Format, werden die Hauptschwerpunkte dieses Kapitels sein.
Anschließend befassen wir uns mit der Generierung sinnvoller Antworten. Dabei werden wir neben einem ELIZA-Programm ein, in einem durch die Möglichkeiten des CPC sehr begrenzten Rahmen lernfähiges, natürlichsprachiges System kennenlernen. Es kann im Dialog mit Ihnen seine Sprachkenntnisse erweitern und auf Ihre Eingaben nicht vorprogrammierte Antworten geben.
Doch bevor wir uns so weit hinauswagen, müssen wir zuerst die Grundlagen erarbeiten. Selbstverständlich können Sie auch hier ständig mit den Demonstrations-Programmen nach Belieben experimentieren.

2.1.1 Die Konversation in natürlicher Sprache

Schauen Sie sich einmal das folgende kurze BASIC-Programm an:

```
10 print
20      input te$
30      print "Ich verstehe."
40 go to 10
```

Eine Konversation mit dem Programm könnte so aussehen:

?ICH MAG MORGENS GERNE LANGE SCHLAFEN
Ich verstehe.

?HEUTE IST FANTASTISCHES WETTER DESHALB FAHRE ICH AN DIE OSTSEE ZUM SURFEN
Ich verstehe.

Das ist wirklich ein absurdes, um nicht zu sagen »idiotisches«, Programm. Obwohl es ständig beteuert »Ich verstehe.«, versteht es nicht ein einziges Wort der Konversation. Das wird auch jemandem, der den Programmaufbau nicht kennt, deutlich, wenn er sich mit dem Programm »unterhält«.
Der sinnvollste Test, mit dem Sie feststellen können, ob ein Programm natürliche Sprache versteht, besteht darin, ihm Fragen über einen bereits eingegebenen Text zu stellen.
Das sähe dann beispielsweise so aus:

?WAS MAG ICH MORGENS GERNE?
Ich verstehe.

?IST DAS WETTER HEUTE FANTASTISCH?
Ich verstehe.

Damit ist das Programm »enttarnt«. Die beschriebene Methode, einem Programm Fragen über einen zuvor eingegebenen Text zu stellen, hilft Ihnen bei jedem Programm schnell, festzustellen, inwieweit es natürliche Sprache wirklich versteht.
Angenommen, das Programm hätte die folgenden Antworten gegeben:

?WAS MAG ICH MORGENS GERNE?
Du magst morgens gerne lange schlafen.

?IST DAS WETTER HEUTE FANTASTISCH?
Ja.

In diesem Fall könnte man durchaus sagen, daß das Programm die Konversation verstünde, da es alle Fragen sinnvoll und korrekt beantwortet hat.
In den letzten 5 bis 10 Jahren sind zu derartigen Leistungen fähige Programme entwickelt worden. Sie haben nur den Nachteil, daß sie alle lediglich ein Spezialgebiet beherrschen. Der Grund dafür liegt in der Tatsache, daß zur Verarbeitung natürlicher Sprache immer Fachwissen benötigt wird.
Ein Beispiel für ein derartiges Programm ist SHRDLU. Es unterhält sich mit dem User munter über eine »Klötzchenwelt«.
Ein Beispieldialog SHRDLUs lautet:

MENSCH:	Würdest du bitte einen Turm aus den beiden blauen Klötzchen oder einem Kegel und dem gelben Klötzchen bauen?
SHRDLU:	O.K.
MENSCH:	Welches Klötzchen liegt ganz unten?
SHRDLU:	Das große gelbe, auf dem der Kegel steht.
MENSCH:	Befindet sich neben dem Turm ein weiterer Würfel?
SHRDLU:	Ja, es stehen zwei neben dem Turm: Ein kleiner roter und ein gelber.

Im Bild 2.1 ist die aus farbigen Klötzchen, Pyramiden und Kisten bestehende Welt von SHRDLU auf einer Tischplatte dargestellt. SHRDLU ist sehr leistungsfähig. Wenn das Programm läuft, erscheint ein Bild auf dem Display des Computers, wie es beispielhaft in der vorherigen Graphik dargestellt ist.
SHRDLU ändert auf Wunsch auch die Positionen der einzelnen Bauklötzchen. Der Benutzer kann dann Informationen über alle beliebigen Einzelheiten der Klötzchenwelt erfragen. Der kurz vorgestellte Dialog gibt dabei nur einen Bruchteil der Möglichkeiten des Programms wieder. SHRDLU setzte mit seiner Leistungsfähigkeit einen richtungweisenden Meilenstein in der KI-Forschung.

2.1.2 Realisierung eines Pseudo-Dialogs mit dem CPC

Auch wir werden uns bei der Kommunikation mit dem CPC nur mit speziellen Sachgebieten befassen. Zunächst stelle ich Ihnen das Programm DOC vor. Dieses Psychiaterprogramm ist zu einem Pseudo-Dialog fähig. Dabei täuscht es dem Benutzer quasi Verständnis vor. Es stellt in

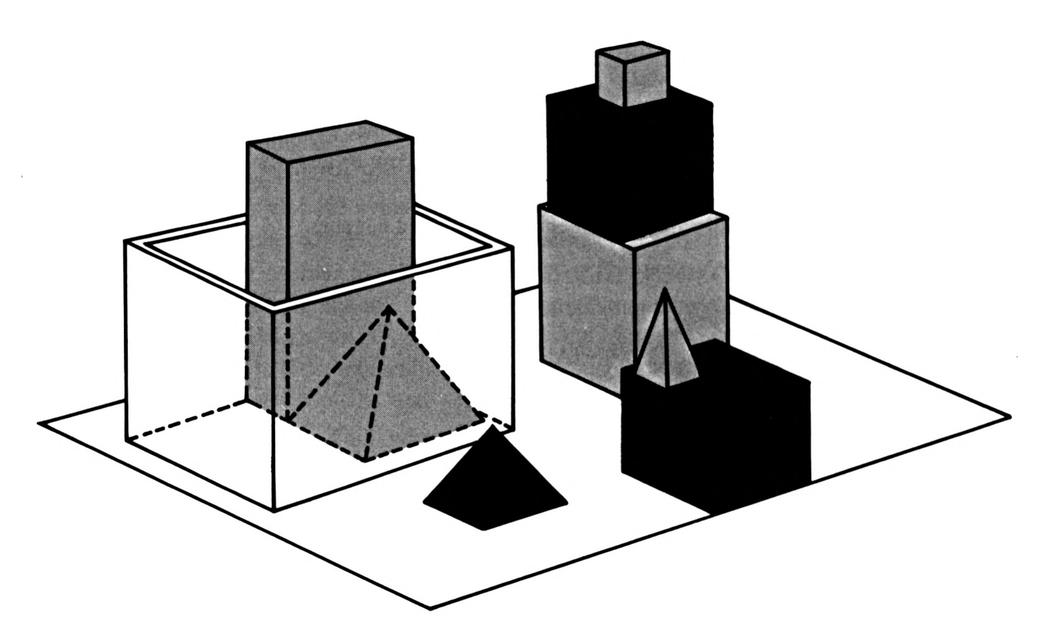


Bild 2.1

seiner Konzeption eine einfache Vorstufe zum ELIZA-Programm dar und ist zu dessen Verständnis notwendig.

Das Programm ist, wie alle Programme dieses Buches, aus strukturierten Unterprogrammen, Prozeduren, aufgebaut. Nachfolgend sehen Sie die einzelnen Programm-Module, die Sie nacheinander eingeben sollten.

Der Supervisor, der Programmkopf, sieht folgendermaßen aus:

```
1 REM Kuenstliche Intelligenz auf dem CPC
2 REM Titel : DOC - Pseudodialog
5 :
6   PAPER 0:PEN 1
7   MODE 1
8   BORDER 9,9
10 :
30   CLS
35   WIDTH 60
40 :
100 REM Initialisation
110 :     GOSUB 1000
200 REM Einleitung
210 :     GOSUB 2000
300 REM Dialog
310 :     GOSUB 3000
400 REM Ende
410 :     GOSUB 4000
500 REM neuer Durchgang?
510 :     GOSUB 5000
600 STOP
```

Der Programmkopf ruft die fünf Unterroutinen »Initialisation«, »Einleitung«, »Dialog«, »Ende« und »neuer Durchgang« auf. Die einzelnen Prozeduren lauten folgendermaßen:

Der Initialisationsteil

Die Routine definiert zunächst ein Textfeld »t$« mit 20 Feldern. Anschließend werden die einzelnen Felder mit Kommentaren oder Fragen von DOC belegt:

```
1000 REM Initialisation
1001   PRINT"DOC-PSEUDODIALOG    (c) OLAF HARTWIG"
1002   WINDOW 2,40,2,40
1003    PEN 7
1004    PAPER 5
1008   CLS
1009 :
1010 DIM t$(20)
1020    FOR i=1 TO 20
1030      READ t$(i)
1040    NEXT i
1050 :
1060 RETURN
1070 :
1100 REM Vokabular
1110    DATA "Glaubst du, das ist normal?"
1120    DATA "Schlaefst du gut?"
1130    DATA "Woher kommt das?"
1140    DATA "Warum?"
1150    DATA "Aha! So ist das also."
1160    DATA "Bitte sprich weiter."
1170    DATA "Gibt es dafuer Gruende?"
1180    DATA "Wie hast du heute geschlafen?"
1190    DATA "Bist du oft unzufrieden?"
1200    DATA "Was hast du zum Fruehstueck gegessen?"
1210    DATA "Erzaehle mir mehr."
1220    DATA "Denkst du, du kannst damit leben?"
1230    DATA "Wieviele Freunde hast du?"
1240    DATA "Was haelst du von Geld?"
1250    DATA "Ich verstehe."
1260    DATA "Hast du Angst vor Computern?"
1270    DATA "Wirklich?"
1280    DATA "Bist du sicher?"
1290    DATA "Denkst du so schon lange?"
1300    DATA "Prima!"
```

Die Einleitung

In diesem Programm-Modul wird ein einleitender Text angezeigt und ein zweites Bildschirmfenster gesetzt. Sie werden aufgefordert, Ihren Namen einzugeben, und anschließend beginnt DOC den Pseudo-Dialog:

```
2000 REM Einleitung
2010 :
2020 PRINT "Hallo und guten Tag!"
```

```
2030 PRINT
2040 PRINT "Ich bin Doc und helfe"
2050 PRINT
2055 PRINT "dir bei all deinen Problemen."
2058 PRINT
2060 PRINT "Wie heisst du eigentlich"
2070   INPUT n$
2080 :
2100   PRINT:PRINT
2105   WINDOW 5,40,7,40
2106   PAPER 2
2107    PEN 3
2108      CLS
2110         PRINT "Ok, ";n$;" was bedrueckt dich?"
2120          INPUT "----> ";a$
2130      PRINT
2200 RETURN
```

Der Dialogteil

Dies ist das Kernstück des Programms. In zwölf Durchgängen wird aus dem Vokabular ein zufälliger Kommentar ausgewählt. Sie müssen hier auf die Frage oder Bemerkung antworten. Leere Eingaben erzeugen eine Fehlermeldung und eine Aufforderung zur neuen Eingabe:

```
3000 REM Dialog
3010  FOR i=1 TO 12
3020      REM 12 Durchgaenge
3030        a=INT(20*RND(i))+1
3040          IF z1=a THEN 3030
3050             z1=a
3060             a$=""
3065          PRINT "....> ";t$(a)
3070          INPUT "----> ";a$
3072         IF a$="" THEN PRINT "Antworte meinen Fragen.":
             GOTO 3070
3080        PRINT
3090  NEXT i
3095 :
3100 RETURN
```

Der Schlußteil

Nach Beendigung der zwölf Durchgänge wird eine abschließende Bemerkung angezeigt:

```
4000 REM Ende
4010   PRINT
4020    PRINT
4030     PRINT "Prima, du hast bereits grosse"
4040     PRINT "Fortschritte bei der Bewaeltigung"
4050     PRINT "Deiner Probleme gemacht ...."
4100 RETURN
```

Die Wiederholungsabfrage

Hier haben Sie die Möglichkeit, eine neue Session zu wählen oder den Programmablauf zu beenden.

```
5000 REM Wiederholung?
5010  PRINT
5020   PRINT
5030     PRINT "Wuenschst du noch eine Session?(J/ )";
5040     INPUT s$
5050   IF s$="j" THEN RUN
5060  PRINT "Dann eben nicht!"
5070  PRINT "Bis zum naechsten Mal..."
5080 PRINT
5090 RETURN
```

Ein Demonstrationslauf von DOC

Nachdem das Programm mit RUN gestartet wurde, kann ein Beispieldialog folgendermaßen aussehen:

Hallo und guten Tag!
Ich bin DOC und helfe
dir bei all deinen Sorgen.
DOC: Wie heißt du eigentlich?
––> Stephan
DOC: Ok, Stephan, was bedrückt dich?
––> Ich fühle mich nicht gut.
DOC: Warum?
––> Ich schlafe sehr schlecht in letzter Zeit.
DOC: Glaubst du, das ist normal?
––> Sicher nicht.
DOC: Denkst du, du kannst damit leben?
––> Nein, deshalb bin ich ja bei Ihnen.
DOC: Bist du sicher?
––> Absolut!
.
.
.

Prima, du hast bereits große
Fortschritte bei der Bewältigung
deiner Probleme gemacht ...

Es macht zwar Spaß, sich mit DOC zu unterhalten, gleichzeitig sollte aber deutlich geworden sein, was ein Pseudo-Dialog ist. DOC ist nicht wirklich intelligent, sondern besitzt ein so allge-

mein abgefaßtes Vokabular, daß die zufällig gewählte Bemerkung oder Frage zu fast allen vom Dialogpartner eingegebenen Bemerkungen paßt.

In der Forschung entwickelte, natürliche Sprache verstehende Programme besitzen oft die Fähigkeit zum Pseudo-Dialog. Wenn ein Programm die Eingabe des menschlichen Benutzers nicht versteht, greift es auf einen »schwammigen« Wortschatz wie den von DOC zurück und erweckt mit diesen »Sprechblasen« beim Anwender den Eindruck des Verstehens.

Am DOC-Programm läßt sich auch das Grundprinzip eines jeden sprachverstehenden Systems verdeutlichen: Notwendig ist immer ein Grundvokabular, aus dem sinnvolle Antwortsätze generiert werden können. Gleichzeitig müssen Ihre Bemerkungen angenommen werden. Zu einem wirklichen Dialog gehört aber auch das »Verstehen« dieser Bemerkungen. Dazu wird Fachwissen benötigt.

Das Beispiel von DOC wurde vor allem deshalb gewählt, um Ihnen zu zeigen, wo die Schwierigkeiten beim Verstehen natürlicher Sprache liegen. DOC verfügt nicht über das benötigte Fachwissen. Dieses Wissen werden wir in einem begrenzten Rahmen bei späteren Programmen implementieren. Um einen wirklich sinnvollen Dialog zu ermöglichen, ist es an dieser Stelle sinnvoll, bei den Grundlagen anzufangen: Wir werden im folgenden klären, was natürliche Sprache eigentlich ist, und uns dann überlegen, wie der Computer die vom Benutzer getätigten Eingaben analysieren und deren Syntax bzw. Bedeutung erkennen kann. Diesen Vorgang der Analyse nennt man »Parsing«.

In den folgenden Abschnitten des Buches wird stufenweise ein komplettes Parsing-Programm entwickelt. Es besitzt in der Endversion sogar die Fähigkeit zum Lernen und zur Erweiterung des Parsing-Vokabulars.

Anschließend werden wir uns mit der Generierung sinnvoller Antworten befassen und aufbauend auf diesem Wissen MICRO-DBABY entwickeln, ein Programm, das fähig ist, seinen Wortschatz im Dialog mit Ihnen zu vergrößern und Ihnen »sinnvolle« Antworten zu geben.

2.2 Was ist natürliche Sprache?

Über diese Frage sind schon sehr viele Abhandlungen und Bücher geschrieben worden. Unter Sprachwissenschaftlern scheint sich die Definition Noam Chomskys durchzusetzen, der Sprache als eine Abfolge von einzelnen Sätzen definiert. Jeder Satz stellt dabei eine Information dar. Uns bringt diese Definition wenig weiter.

Für die Entwicklung unserer Programme ist es sinnvoller, Sprache vorerst als eine Aneinanderreihung einzelner Wörter zu sehen. Wir haben die Möglichkeit, Sätze in die einzelnen Wörter zu zerlegen. Diese Wörter können dann später zu grammatisch korrekten Antworten synthetisiert werden. Wenden wir dieses Wissen einmal im folgenden Parsing-Programm an.

2.2.1 Parsing: Die Dekodierung von Sätzen

Zunächst muß der CPC einen eingegebenen Satz erkennen. Wie erkennen Sie einen Satz? Nun, sehr einfach, an seinen Satzzeichen! Haben Sie schon einmal versucht, einen Text ohne jegliche Interpunktion zu lesen? Für einen Computer ist es einfach, in einem eingegebenen Text nach einem Punkt, einem Ausrufezeichen oder einem Fragezeichen zu suchen.
Wir werden nun das Parsing-Programm entwickeln, das den eingegebenen Text zunächst in Sätze, dann in die einzelnen Wörter zerlegt. Anschließend werden die Satzteile bestimmt, um so später auf die Syntax schließen zu können.
Der Programmkopf des Dekodier-Programms sieht folgendermaßen aus:

```
4 REM Titel : Parsingprogramm
5 :
6 MODE 1
15 PEN 1
20 PAPER 0:CLS
40 :
100 GOSUB 1000
110     REM Initialisation
200 GOSUB 2000
210     REM Eingabe
250   WINDOW 1,40,8,25
260   PAPER 7
270   CLS
280 :
300 GOSUB 3000
310     REM Satzzeichen suchen
400 GOSUB 4000
410     REM Satzzuweisung
500 GOTO 300
600 STOP
```

Der Steuerungsteil wird später noch erweitert. Zunächst muß im Unterprogramm »INITIALISATION« ein Feld definiert werden, in dem die einzelnen Sätze des einzugebenden Textes abgelegt werden können. Der Satz-Pointer »s« wird auf Eins, d.h. das erste Satzfeld gesetzt. Anschließend erfolgt die Anzeige des Titels:

```
1000 REM Initialisation
1010    DIM s$(10)
1020      REM 10 Saetze dimensionieren
1030    s=1
1040      REM Satzzaehler reset
1100 CLS
1110 PRINT
1115 :
1150   PRINT "Bitte geben Sie Ihre Saetze ein...>"
1160   PRINT
1500 RETURN
```

Der Eingabeteil des Parsing-Programms

Der Programmteil stellt eine nützliche Unterroutine dar, die es ermöglicht, Texte auch mit in der normalen INPUT-Eingabe unerlaubten Sonderzeichen einzugeben. Eingegebener Text wird mit <RETURN> beendet. Die Korrekturen erfolgen mittels der <DEL>-Taste.
Dieser Programmteil lautet:

```
2000 REM Eingabe
2005     te$=""
2007     t$="":LOCATE 1,4:PRINT te$;
2010       t$=INKEY$
2015       IF t$="" THEN 2010
2020         PRINT t$;
2030           IF t$=CHR$(13) THEN RETURN
2100           IF t$=CHR$(242) AND LEN(te$)>1 THEN te$=LEFT$(te$,
               LEN(te$)-1):PRINT" ";:GOTO 2007
2110         te$=te$+t$
2120       GOTO 2010
```

Die Suche nach Satzzeichen

Der unter der Variablen »te$« abgespeicherte Text wird Zeichen für Zeichen in einer FOR-NEXT-Schleife nach den Satzzeichen Punkt, Ausrufezeichen und Fragezeichen durchsucht. Das aktuelle Zeichen wird unter der Variablen »su$« gespeichert und mit den drei möglichen Satzzeichen verglichen:

```
3000 REM Satzzeichen suchen
3010     FOR i=1 TO LEN(te$)-1
3020        su$=MID$(te$,i,1)
3030            IF su$="." OR su$="!" OR su$="?" THEN RETURN
3050     NEXT i
3990 :
3995 RETURN
```

Die Zuweisung einzelner Sätze

Ist in der obigen Routine ein Satzzeichen gefunden worden, legt das hier folgende Unterprogramm den Text vom Textbeginn bis zu diesem Zeichen in dem Satzfeld »s$(s)« ab. Anschließend wird der Satz-Pointer »s« um einen Wert erhöht.
In den Zeilen 4300 ff. wird der in dem Feld »s$(s)« abgelegte Satz von dem insgesamt unter »te$« gespeicherten Text der Eingabe abgetrennt. Der so entstandene Resttext wird daraufhin untersucht, ob er leer ist, ob also alle Sätze zugewiesen sind. Ist dies der Fall, so springt das Programm in die Zeile 600 zu einem vorläufigen »Stop«-Befehl. Später fügen wir hier die weitere Zerlegung der einzelnen Sätze in Wörter ein.
Die Prozedur sieht folgendermaßen aus:

```
4000 REM Satzzuweisung
4100    s$(s)=LEFT$(te$,i)
4140        PRINT
```

```
4150      PRINT "Satz Nr. ";s;" = ";s$(s)
4200   s=s+1
4300   te$=RIGHT$(te$,LEN(te$)-i)
4350       PRINT "Resttext..Nr.";s;"...";te$
4400   IF te$="" THEN 600
4500 RETURN
```

Die Dekodierung von Sätzen in Wörter

Wir haben jetzt einen eingegebenen Text in einzelne Sätze zerlegt. Als nächstes müssen wir uns mit der Isolierung der Satzteile befassen, um diese dann später syntaktisch analysieren zu können. Fügen Sie zunächst im Initialisationsteil in der Zeile 1000 ff. die folgenden Anweisungen ein:

```
1100 DIM w$(10,20)
1100 REM              Wortfeld max. 10 Sätze mit je bis zu 20 Wörtern.
```

Überschreiben Sie die Zeile 600 und erweitern Sie den Supervisor um folgende Zeilen:

```
600 FOR w=1 TO s
605    REM w.ten Satz in die einzelnen Woerter aufteilen
610      GOSUB 10000
620      REM Woerter isolieren
650 NEXT w
700 PRINT:PRINT "Alle Woerter sind in das w$(x,y)-Feld
    aufgeteilt.":PRINT:STOP
```

Fügen Sie anschließend die neue Wortisolierungsroutine in den Zeilen 10.000 ff. an:

```
10000 REM w.ten Satz in Woerter teilen
10030   se=1
10040   i=0
10050   wo$=s$(w)
10060    IF LEFT$(wo$,1)=" " THEN wo$=RIGHT$(wo$,LEN(wo$)-1)
10100    i=i+1
10130      sp$=MID$(wo$,i,1)
10150       IF sp$=" " THEN GOSUB 11000
10160       REM Leerzeichen gefunden
10200    IF i<=LEN(wo$) THEN 10100
10210  GOSUB 11000
10215 RETURN
10990 :
11000 REM Wort isolieren
11010   w$(w,se)=LEFT$(wo$,i)
11020   i=i-LEN(w$(w,se))
11030      se=se+1
11040      REM Satzelementnummer erhoehen
11100         wo$=RIGHT$(wo$,LEN(wo$)-LEN(w$(w,se-1)))
11110         REM Restsatz festlegen
11200 RETURN
```

Das Unterprogramm besteht aus drei Teilen. Zuerst werden die lokalen Variablen »se«, »i« und »wo$« festgelegt. Die Variable »wo$« beinhaltet dabei den momentan zu bearbeitenden Satz. Die

Variable »se« stellt die Nummer des aktuellen Wortes innerhalb des »wo$«-Satzes dar. »I« ist ein Schleifenzähler, der die Position des jeweils auf ein Leerzeichen zu untersuchendes Zeichen des Satzes festlegt. Sollte das erste Zeichen des Satzes in »wo$« ein Leerzeichen sein, so wird es in der Zeile 10060 gelöscht.
Die Zeilen 10100 bis 10200 stellen die Leerzeichen-Suchschleife dar. Wurde ein Leerzeichen gefunden, so springt das Programm in die Unterprozedur in der Zeile 11000 ff. Dort erfolgt die Zuweisung des gefundenen Wortes in das Feld »w$(w,se)«.
Anschließend werden die Satzelement-Nummer erhöht und der Satz-String von dem in das Feld »w$()« abgelegten Wort abgetrennt. Mit dem so erhaltenen Restsatz wird die Isolierung der einzelnen Wörter fortgeführt. Diese Prozedur wiederholt sich für alle Sätze.
Lassen Sie das nun neu erweiterte Dekodierungs-Programm einmal laufen. Geben Sie dazu beispielsweise die folgenden Sätze ein:

> »Heute ist Montag. Es regnet immer noch in Strömen. Wo habe ich meinen Schirm gelassen? Immer verlege ich ihn.«

Das Programm erkennt zunächst die einzelnen vier Sätze und legt sie im Feld »s$()« ab. Das sieht dann folgendermaßen aus:

s$(1)=»Heute ist Montag.«
s$(2)=»Es regnet immer noch in Strömen.«
s$(3)=»Wo habe ich meinen Schirm gelassen?«
s$(4)=»Immer verlege ich ihn.«

Anschließend werden die Sätze in Wörter zerlegt und dem Feld »w$(Satz,Wörter)« zugewiesen:

Satz Nr: W$(Satz,Wort)-Wörter:

	1	2	3	4	5	6
1	Heute	ist	Montag.			
2	Es	regnet	immer	noch	in	Strömen.
3	Wo	habe	ich	meinen	Schirm	gelassen?
4	Immer	verlege	ich	ihn.		

Mit dieser internen Repräsentation der Sprachinhalte ist ein sehr wichtiger Punkt auf dem Weg zur Dekodierung und zum Verstehen natürlicher Sprache durch den Computer gelöst.

2.2.2 Das grammatische Format

Widmen wir uns nun dem nächsten Problem, der grammatikalischen Analyse der einzelnen eingegebenen Sätze. Dazu werden wir das Parsing-Programm erneut erweitern. Die grammatische Analyse ist dabei direkt mit dem letzten Schritt, dem Verstehen der Syntax der Sätze, verbunden. Das Dekodierungs-Programm wird mit der nun vorgestellten Erweiterung in der Lage sein, zu erkennen, wie ein Satz zusammengesetzt ist, d.h. aus welchen Satzteilen er besteht.

Das in unserem Programm beschrittene Verfahren basiert auf dem Grundprinzip, daß der CPC zunächst die Wortart jedes einzelnen Wortes der Eingabe bestimmt. Dies erfolgt über den Vergleich des jeweiligen Wortes mit einem im Programm enthaltenen Grundvokabular. Ausgehend von der Wortart können anschließend Rückschlüsse auf die Satzteile gezogen werden, da Wortarten in der Regel in einem Satz in charakteristischer Reihenfolge auftreten.
Der Satz

»Es regnet in Strömen«

besteht aus

Subjekt, Verb und adverbaler Bestimmung.

Ein Mensch erkennt das sofort. Wie bringen wir aber den Computer dazu, diese Aufgabe zu bewerkstelligen?

2.2.3 Das modifizierte Parsing-Programm

Um die Zuweisung von Satzteilen zu erreichen, muß das Parsing-Programm, wie bereits zuvor erwähnt, geändert werden. Überschreiben Sie die Zeile 700 mit

```
700 for w=1 to s
```

und fügen Sie die folgenden Zeilen hinzu:

```
700 FOR w=1 TO s-1
710     REM w.ten Satz analysieren
720      GOSUB 20000
725 PRINT
726 PRINT"Druecken Sie eine Taste..."
727 q$=INKEY$
728 IF q$="" THEN 727
729 CLS
730 NEXT w
800 PRINT:PRINT
820 PRINT"Ende des Parsingvorgangs!"
830 STOP
```

Dadurch wird der Analyseteil, der später in Zeile 20000 ff. entwickelt wird, für jeden einzelnen Satz aufgerufen.
Zuerst benötigen wir aber eine Wissensbasis, d.h. der Computer muß die Wortarten kennen – sonst könnte er keine Analyse vornehmen. Wie soll er beispielsweise wissen, daß »schön« ein Adjektiv ist oder daß »Haus« in diesem Fall ein Objekt darstellt?
Erweitern Sie dazu den Initialisationsteil:

```
1000 REM initialisation
1010    DIM s$(10)
1020      REM 10 saetze dimensionieren
1030    s=1
1040      REM satzzaehler reset
```

```
1100 DIM w$(10,20)
1101   REM Wortfeld max.10 Saetze mit bis zu 20 Woertern
1105 CLS
1110 :
1115 PRINT "PARSING ___ Analyse von Texten __ nach grammatischer
           Struktur __ (c)Olaf Hartwig";
1120 PRINT
1130 ORIGIN 0,380:DRAWR 660,0
1140 WINDOW 5,75,3,6
1142 PAPER 1:PEN 0
1145 CLS
1150   PRINT "Bitte geben Sie Ihre Saetze ein...>"
1160 :
1200 DIM no$(100)
1210     REM Nomen
1220 DIM ve$(100)
1230     REM Verben
1240 DIM pr$(100)
1250     REM Praepositionen
1280 DIM ar$(100)
1285     REM Artikel
1290 DIM ad$(100)
1295     REM Adjektive
1299 :
1300 REM Satzteile einlesen
1310   FOR i=1 TO 100
1312       READ no$(i)
1313       IF no$(i)="*" THEN 1320
1314   NEXT i
1315 :
1320   FOR i=1 TO 100
1322       READ ve$(i)
1323       IF ve$(i)="*" THEN 1340
1324   NEXT i
1330 :
1340   FOR i=1 TO 100
1342       READ pr$(i)
1343       IF pr$(i)="*" THEN 1350
1344   NEXT i
1345 :
1350   FOR i=1 TO 100
1352       READ ar$(i)
1353       IF ar$(i)="*" THEN 1360
1354   NEXT i
1355 :
1360   FOR i=1 TO 100
1362       READ ad$(i)
1363       IF ad$(i)="*" THEN 1367
1364   NEXT i
1366 :
1367 RETURN
1369 :
1370      REM Nomen
1371       DATA ich,du,er,sie,es,wir
1372       DATA ihr,es, auto,haus,schirm
```

```
1375       DATA regen
1397       DATA *
1400      REM Verben
1405       DATA gehen,regnet,laufen,spiele
1410       DATA eilen,langweilen,fliegen
1415       DATA trinken,schlafen,sausen
1445       DATA *
1450      REM Praepositionen
1455       DATA ueber,unter,neben
1457       DATA bei,an,auf
1458       DATA *
1460      REM Artikel
1465       DATA der,die,das
1467       DATA ein,eine
1468       DATA *
1470      REM Adjektive
1475       DATA schoen,gruen,hoch
1480       DATA aufregend,beindruckend
1485       DATA toll,fantastisch
1490       DATA schnell,rasant,intensiv
1498       DATA *
1499 :
1500 RETURN
```

Die Programmerweiterung liest die folgenden Wortarten in die dazugehörigen Felder:

Wortart:	Feld:
Substantive	no$()
Verben	ve$()
Präpositionen	pr$()
Artikel	ar$()
Adjektive	ad$()

Sie können die DATAs dieser Wortarten-Felder in den Zeilen 1370 bis 1498 beliebig erweitern, da die maximale Feldlänge nur durch die Speicherkapazität des Computers begrenzt ist. Enthält ein Feld mehr als 100 Daten, so müssen lediglich die DIM-Befehle in den Zeilen 1200 ff. geändert werden. Das Programm bestimmt die Anzahl der Feldelemente selbst durch den Feldende-Marker »*«.
Damit haben Sie eine kleine Wissensbasis errichtet. Geben Sie nun die folgende Prozedur zur eigentlichen grammatischen Analyse ein:

```
20000 REM grammatische Analyse
20005 PRINT
20010   REM Beseitigung von Leerzeichen und Satzzeichen
20020    FOR x=1 TO 10
20030      y=w
20040         IF w$(y,x)<>"" THEN w$(y,x)=LEFT$(w$(y,x),
              LEN(w$(y,x))-1)
20060       NEXT x
20070 :
```

```
20100 REM Wort selektieren
20105  se=1
20110  wo$=w$(w,se)
20120    GOSUB 25000
20130    REM Vergleich
20140    se=se+1
20150   IF w$(w,se)="" THEN RETURN
20160   REM Satz beenden
20170 GOTO 20110
24990 :
25000 REM Vergleich
25090 :
25100 REM Nomen?
25105   vg$="Nomen"
25110     FOR i=1 TO 100
25120       IF no$(i)=wo$ THEN GOTO 29000
25130       IF no$(i)="*" THEN 25200
25150     NEXT i
25190 :
25200 REM Verb?
25205   vg$="Verb"
25210     FOR i=1 TO 100
25220       IF ve$(i)=wo$ THEN GOTO 29000
25230       IF ve$(i)="*" THEN 25300
25250     NEXT i
25290 :
25300 REM Praeposition?
25305   vg$="Praeposition"
25310     FOR i=1 TO 100
25320       IF pr$(i)=wo$ THEN GOTO 29000
25330       IF pr$(i)="*" THEN 25400
25350     NEXT i
25390 :
25400 REM Artikel?
25405   vg$="Artikel"
25410     FOR i=1 TO 100
25420       IF ar$(i)=wo$ THEN GOTO 29000
25430       IF ar$(i)="*" THEN 25500
25450     NEXT i
25490 :
25500 REM Adjektiv?
25505   vg$="Adjektiv"
25510     FOR i=1 TO 100
25520       IF ad$(i)=wo$ THEN GOTO 29000
25530       IF ad$(i)="*" THEN 25600
25550     NEXT i
25580 IF wo$=" " OR wo$="" THEN RETURN
25590 :
25600 PRINT "Das Wort ";wo$;" ist nicht bekannt!"
25620 RETURN
29000 PRINT "Das Wort ";wo$;" ist ein ";vg$
29010 RETURN
```

Wie läuft die im obigen Programm-Modul realisierte grammatische Analyse ab?
Das Programm muß zunächst die im Feld »w$(Satz,Wörter)« enthaltenen Leerzeichen und Satzzeichen entfernen. Werden die einzelnen Wörter in diesem Feld abgelegt, so befindet sich hinter jedem Wort noch ein Leerzeichen bzw. hinter dem letzten Wort des Satzes noch das Satzzeichen. Der Satz

»Ich_fühle_mich_gut!«

wird beispielweise folgendermaßen abgespeichert:

»Ich_« , »fühle_« , »mich_« und »gut!«.

Die hier mit »_« dargestellten Sonderzeichen werden in den Zeilen 20010 bis 20060 gelöscht. Anschließend wird der Variablen „wo$" ein Wort des momentan zu untersuchenden Satzes zugewiesen. Dieses Wort wird nun mit allen Wortart-Feldern verglichen. Das Programm gibt dann aus, zu welcher Wortart das untersuchte Wort gehört. Natürlich können nur bekannte, also in den Wortart-Feldern abgelegte Wörter verglichen werden, andernfalls gibt das Programm zu verstehen, daß es die Wortart noch nicht kennt.
Hier ist es sinnvoll, die Wortart-Felder nach der oben beschriebenen Methode zu erweitern. Damit wird auch schon ansatzweise das größte Problem der meisten KI-Algorithmen deutlich:

> Fast immer ist zu einer Problemlösung eine große Wissensbank nötig. Und je umfangreicher dieses Wissen ist, desto komplexer wird der Programmteil, der das Wissen verwaltet. Zur Verwaltung ist wieder zusätzliches Wissen bzw. sind Regeln nötig, die den Umfang der Wissensbank weiter vergrößern. Fast jedes leistungsfähige KI-Programm beansprucht daher einen sehr großen, oft im Megabyte-Bereich liegenden Speicherbereich.

Bei unserem Programm stößt man mit dem CPC natürlich schnell an die Grenzen der Speicherkapazität. Es dient vor allem dazu, Ihnen die Grundprinzipien der Sprachanalyse sowie deren Probleme und Grenzen aufzuzeigen. Gleichzeitig können Sie das Dekodierungs-Programm beliebig verändern und damit experimentieren.
Unten sehen Sie das komplette Parsing-Programm als Listing mit dem neuen Analyse- und Initialisationsteil sowie einer ganzen Reihe von Detailoptimierungen:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel : Parsing
5 :
6 MODE 2
7 BORDER 9,9
8 WIDTH 60
10 :
20 PAPER 0
25 PEN 1
27 :
30 CLS
40 :
100 GOSUB 1000
110      REM Initialisation
200 GOSUB 2000
```

```
210      REM Eingabe
220 :
250 WINDOW 1,80,8,25
260 PAPER 7
270 CLS
275 PEN 0
280 :
300 GOSUB 3000
310      REM Satzzeichen suchen
400 GOSUB 4000
410      REM Satzzuweisung
500 GOTO 300
520 :
600 FOR w=1 TO s
605     REM w.ten Satz in einzelne Woerter aufteilen
610       GOSUB 10000
620       REM Woerter isolieren
650 NEXT w: PRINT
655 PRINT "Druecken Sie eine Taste zum Parsingvorgang."
656 q$=INKEY$:IF q$="" THEN 656
660 :
680 WINDOW 30,79,10,25
690  PAPER 0
692  PEN 1
694 CLS
696 :
700 FOR w=1 TO s-1
710      REM w.ten Satz analysieren
720       GOSUB 20000
725 PRINT
726 PRINT"Druecken Sie eine Taste..."
727 q$=INKEY$
728 IF q$="" THEN 727
729 CLS
730 NEXT w
800 PRINT:PRINT
820 PRINT"Ende des Parsingvorgangs!"
830 STOP
900 :
1000 REM Initialisation
1010     DIM s$(10)
1020       REM 10 Saetze dimensionieren
1030     s=1
1040       REM Satzzaehler reset
1100 DIM w$(10,20)
1101    REM Wortfeld max.10 Saetze mit bis zu 20 Woertern
1105 CLS
1110 :
1115 PRINT "PARSING ___ Analyse von Texten __ nach grammatischer
           Struktur __ (c)Olaf Hartwig";
1120 PRINT
1130 ORIGIN 0,380:DRAWR 660,0
1140 WINDOW 5,75,3,6
1142 PAPER 1:PEN 0
1145 CLS
1150    PRINT "Bitte geben Sie Ihre Saetze ein...>"
```

```
1160 :
1200 DIM no$(100)
1210      REM Nomen
1220 DIM ve$(100)
1230      REM Verben
1240 DIM pr$(100)
1250      REM Praepositionen
1280 DIM ar$(100)
1285      REM Artikel
1290 DIM ad$(100)
1295      REM Adjektive
1299 :
1300 REM Satzteile einlesen
1310   FOR i=1 TO 100
1312       READ no$(i)
1313       IF no$(i)="*" THEN 1320
1314   NEXT i
1315 :
1320   FOR i=1 TO 100
1322       READ ve$(i)
1323       IF ve$(i)="*" THEN 1340
1324   NEXT i
1330 :
1340   FOR i=1 TO 100
1342       READ pr$(i)
1343       IF pr$(i)="*" THEN 1350
1344   NEXT i
1345 :
1350   FOR i=1 TO 100
1352       READ ar$(i)
1353       IF ar$(i)="*" THEN 1360
1354   NEXT i
1355 :
1360   FOR i=1 TO 100
1362       READ ad$(i)
1363       IF ad$(i)="*" THEN 1367
1364   NEXT i
1366 :
1367 RETURN
1369 :
1370      REM Nomen
1371       DATA ich,du,er,sie,es,wir
1372       DATA ihr,es, auto,haus,schirm
1375       DATA regen
1397       DATA *
1400      REM Verben
1405       DATA gehen,regnet,laufen,spiele
1410       DATA eilen,langweilen,fliegen
1415       DATA trinken,schlafen,sausen
1445       DATA *
1450      REM Praepositionen
1455       DATA ueber,unter,neben
1457       DATA bei,an,auf
1458       DATA *
1460      REM Artikel
1465       DATA der,die,das
```

```
1467         DATA ein,eine
1468         DATA *
1470        REM Adjektive
1475         DATA schoen,gruen,hoch
1480         DATA aufregend,beindruckend
1485         DATA toll,fantastisch
1490         DATA schnell,rasant,intensiv
1498         DATA *
1499 :
1500 RETURN
1600 :
2000 REM Eingabe
2005    te$=""
2007    t$="":LOCATE 1,2:PRINT te$;
2010      t$=INKEY$
2015      IF t$="" THEN 2010
2020        PRINT t$;
2030          IF t$=CHR$(13) THEN RETURN
2100          IF t$=CHR$(242) AND LEN(te$)>1 THEN te$= LEFT$
              (te$,LEN(te$)-1):PRINT" ";:GOTO 2007
2110        te$=te$+t$
2120      GOTO 2010
2990 :
3000 REM Satzzeichen suchen
3010    FOR i=1 TO LEN(te$)-1
3020       su$=MID$(te$,i,1)
3030           IF su$="." OR su$="!" OR su$="?" THEN RETURN
3050    NEXT i
3990 :
3992 RETURN
3995 :
4000 REM Satzzuweisung
4100   s$(s)=LEFT$(te$,i)
4140      PRINT
4150      PRINT "Satz Nr. ";s;" = ";s$(s)
4200   s=s+1
4300   te$=RIGHT$(te$,LEN(te$)-i)
4350        PRINT "Resttext..Nr.";s;"...";te$
4400   IF te$="" THEN 600
4500 RETURN
7000 :
10000 REM w.ten Satz in Woerter aufteilen
10030   se=1
10040   i=0
10050   wo$=s$(w)
10060    IF LEFT$(wo$,1)=" " THEN wo$=RIGHT$(wo$,LEN(wo$)-1)
10100    i=i+1
10130      sp$=MID$(wo$,i,1)
10150       IF sp$=" " THEN GOSUB 11000
10160       REM Leerzeichen gefunden
10200    IF i<=LEN(wo$) THEN 10100
10210   GOSUB 11000
10215 RETURN
10990 :
11000 REM Wort isolieren
11010   w$(w,se)=LEFT$(wo$,i)
```

```
11020   i=i-LEN(w$(w,se))
11030      se=se+1
11040      REM Satzelement Nummer erhoehen
11100         wo$=RIGHT$(wo$,LEN(wo$)-LEN(w$(w,se-1)))
11110         REM Restsatz festlegen
11200 RETURN
11300 :
20000 REM grammatische Analyse
20005 PRINT
20010   REM Beseitigung von Leerzeichen und Satzzeichen
20020    FOR x=1 TO 10
20030      y=w
20040         IF w$(y,x)<>"" THEN w$(y,x)=LEFT$(w$(y,x),
              LEN(w$(y,x))-1)
20060     NEXT x
20070 :
20100 REM Wort selektieren
20105  se=1
20110  wo$=w$(w,se)
20120    GOSUB 25000
20130    REM Vergleich
20140    se=se+1
20150   IF w$(w,se)="" THEN RETURN
20160   REM Satz beenden
20170 GOTO 20110
24990 :
25000 REM Vergleich
25090 :
25100 REM Nomen?
25105   vg$="Nomen"
25110     FOR i=1 TO 100
25120       IF no$(i)=wo$ THEN GOTO 29000
25130       IF no$(i)="*" THEN 25200
25150     NEXT i
25190 :
25200 REM Verb?
25205   vg$="Verb"
25210     FOR i=1 TO 100
25220       IF ve$(i)=wo$ THEN GOTO 29000
25230       IF ve$(i)="*" THEN 25300
25250     NEXT i
25290 :
25300 REM Praeposition?
25305   vg$="Praeposition"
25310     FOR i=1 TO 100
25320       IF pr$(i)=wo$ THEN GOTO 29000
25330       IF pr$(i)="*" THEN 25400
25350     NEXT i
25390 :
25400 REM Artikel?
25405   vg$="Artikel"
25410     FOR i=1 TO 100
25420       IF ar$(i)=wo$ THEN GOTO 29000
25430       IF ar$(i)="*" THEN 25500
25450     NEXT i
25490 :
```

```
25500 REM Adjektiv?
25505   vg$="Adjektiv"
25510     FOR i=1 TO 100
25520       IF ad$(i)=wo$ THEN GOTO 29000
25530       IF ad$(i)="*" THEN 25600
25550     NEXT i
25580 IF wo$=" " OR wo$="" THEN RETURN
25590 :
25600 PRINT "Das Wort ";wo$;" ist nicht bekannt!"
25620 RETURN
29000 PRINT "Das Wort ";wo$;" ist ein ";vg$
29010 RETURN
```

Ein Parsing-Demonstrationslauf

Das Programm ist zwar etwas länger geworden, doch der Aufwand bei der Eingabe lohnt sich. Wenn Sie den Parser mit RUN starten, erhalten Sie in etwa den folgenden Programmablauf:

PARSING-Analyse von Texten nach grammatischer Struktur

Bitte geben Sie Ihre Sätze ein...>

es regnet. muss das sein? warum habe ich meine jacke zu hause gelassen?

Satz Nr. 1 = es regnet.
Resttext..Nr. 2 ... muss das sein? warum habe ich meine jacke zu hause gelassen?

Satz Nr. 2 = muss das sein?
Resttext..Nr. 2 .. warum habe ich meine jacke zu hause gelassen?

Satz Nr. 3 = warum habe ich meine jacke zu hause gelassen?

Drücken Sie eine Taste zum Parsingvorgang.

Das Wort es ist ein Nomen
Das Wort regnet ist ein Verb.
Das Wort muss ist nicht bekannt!
Das Wort das ist ein Artikel.
Das Wort sein ist nicht bekannt.
.
.
.

2.2.4 Parsing mit automatischem Lernen

Das Parsing-Programm der vorherigen Seiten läßt sich noch entscheidend verbessern. In der Grundversion besitzt das Programm ein geringes Grundvokabular zur Wortartbestimmung. Dieses ist für ernsthaftere Analysen nicht ausreichend. Es ist daher erforderlich, in das Programm eine

Option zu integrieren, die es ermöglicht, das Grundvokabular durch automatisches Lernen zu erweitern.
Das so modifizierte Parsing-Programm folgt auf den nächsten Seiten. Ein beispielhafter Demonstrationslauf ist im folgenden dargestellt.

PARSING-Analyse von Texten nach grammatischer Struktur

Bitte geben Sie Ihre Sätze ein...>

der Satz wird nun analysiert. dies ist ein beispiel fuer den parsingvorgang mit anschliessendem lernmodus.

Satz Nr. 1 = der satz wird nun analysiert.
Resttext Nr. 2 ... dies ist ein beispiel fuer den parsingvorgang mit anschliessendem lernmodus.

Satz Nr. 2 = dies ist ein beispiel fuer den parsingvorgang mit anschliessendem lernmodus.

Drücken Sie eine Taste zum Parsingvorgang.

Das Wort der ist ein Artikel.
Das Wort satz ist nicht bekannt!
Das Wort wird ist nicht bekannt!
Das Wort nun ist nicht bekannt!
Das Wort analysiert ist nicht bekannt!

Drücken Sie eine Taste zum Vok.def.

Legen Sie den grammatischen Satzteil durch Tastendruck fest...

T – Artikel
V – Verb
N – Nomen
A – Adjektiv
P – Präposition

1 satz ...> n OK
2 wird ...> vOK
3 nun ...> p OK
4 analysiert ...> v OK

Das komplette Listing des lernfähigen Parsing-Programms lautet:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel : Parsing mit automatischem Vok.Lernen
5 :
6 MODE 2
7 BORDER 9,9
8 WIDTH 60
10 :
20 PAPER 0
25 PEN 1
```

```
27 :
30 CLS
40 :
100 GOSUB 1000
110     REM Initialisation
200 GOSUB 2000
210     REM Eingabe
220 :
250    WINDOW 1,80,8,25
260    PAPER 7
270     CLS
275    PEN 0
280 :
300 GOSUB 3000
310     REM Satzzeichen suchen
400 GOSUB 4000
410     REM Satzzuweisung
420 :
500 GOTO 300
520 :
600 FOR w=1 TO s
605    REM w.ten Satz in die einzelnen Woerter aufteilen
610      GOSUB 10000
620      REM Woerter isolieren
650 NEXT w:PRINT
652 :
655 PRINT"Druecken Sie eine Taste zum  Parsingvorgang."
656 q$=INKEY$:IF q$="" THEN 656
660 :
662 WINDOW 13,28,10,15
665  PAPER 0
668  PEN 1
669 CLS
670    PRINT  "Legen   Sie  dasgrammatische   Satzteil
               durchTastendruck     fest:"
672    WINDOW 15,28,17,24: CLS: PRINT " N __ Nomen"
673    PRINT " T __ Artikel"
674    PRINT:PRINT " V __ Verb"
675    PRINT
676    PRINT " A __ Adjektiv";
678    PRINT " P __ Praepos.";
680   WINDOW 30,79,10,25
694 CLS
696 :
700 FOR w=1 TO s-1
710     REM w.ten Satz analysieren
720      GOSUB 20000
730 NEXT w
750 :
800 GOTO 40000
950 :
1000 REM Initialisation
1005   DIM nichtbek$(10):REM nicht bekanntes Vokabular
1010    DIM s$(10)
1020      REM 10 Saetze dimensionieren
1030    s=1
```

```
1040       REM Satzzaehler reset
1100 DIM w$(10,20)
1101   REM Wortfeld max.10 Saetze mit bis zu 20 Woertern
1105 CLS
1110 :
1115 PRINT "PARSING ___ Analyse von Texten ___ mit automatischem
           Lernen ___ (c) Olaf Hartwig";
1120   PRINT
1130    ORIGIN 0,380:DRAWR 660,0
1140     WINDOW 1,70,3,6
1142     PAPER 1:PEN 0
1145     CLS
1147 :
1150   PRINT "Bitte geben Sie Ihre Saetze ein...>"
1160 :
1200 DIM no$(100)
1210     REM Nomen
1220 DIM ve$(100)
1230     REM Verben
1240 DIM pr$(100)
1250     REM Praepositionen
1280 DIM ar$(100)
1285     REM Artikel
1290 DIM ad$(100)
1295     REM Adjektive
1299 :
1300 REM Satzteile einlesen
1310   FOR i=1 TO 100
1312       READ no$(i)
1313       IF no$(i)="*" THEN no=i:GOTO 1320
1314   NEXT i
1315 :
1320   FOR i=1 TO 100
1322       READ ve$(i)
1323       IF ve$(i)="*" THEN ve=i:GOTO 1340
1324   NEXT i
1330 :
1340   FOR i=1 TO 100
1342       READ pr$(i)
1343       IF pr$(i)="*" THEN pr=i:GOTO 1350
1344   NEXT i
1345 :
1350   FOR i=1 TO 100
1352       READ ar$(i)
1353       IF ar$(i)="*" THEN ar=i:GOTO 1360
1354   NEXT i
1355 :
1360   FOR i=1 TO 100
1362       READ ad$(i)
1363       IF ad$(i)="*" THEN ad=i:GOTO 1367
1364   NEXT i
1366 :
1367 RETURN
1369 :
1370      REM Nomen
1371       DATA ich,du,er,sie,es,wir
```

```
1372         DATA ihr,es, auto,haus,schirm
1375         DATA regen
1397         DATA *
1400        REM Verben
1405         DATA gehen,regnet, laufen,spiele
1410         DATA eilen,langweilen,fliegen
1415         DATA trinken,schlafen,sausen
1445         DATA *
1450        REM Praepositionen
1455         DATA ueber,unter,neben
1457         DATA bei,an,auf
1458         DATA *
1460        REM Artikel
1465         DATA der,die,das
1467         DATA ein,eine
1468         DATA *
1470        REM Adjektive
1475         DATA schoen,gruen,hoch
1480         DATA aufregend,beindruckend
1485         DATA toll,fantastisch
1490         DATA schnell,rasant,intensiv
1498         DATA *
1499 :
1500 RETURN
1600 :
2000 REM Eingabe
2005    te$=""
2007    t$="":LOCATE 1,2:PRINT te$;
2010      t$=INKEY$
2015      IF t$="" THEN 2010
2020        PRINT t$;
2030          IF t$=CHR$(13) THEN RETURN
2100          IF t$=CHR$(242) AND LEN(te$)>1 THEN te$=LEFT$
              (te$,LEN(te$)-1):PRINT" ";:GOTO 2007
2110        te$=te$+t$
2120      GOTO 2010
2990 :
3000 REM Satzzeichen suchen
3010    FOR i=1 TO LEN(te$)-1
3020       su$=MID$(te$,i,1)
3030           IF su$="." OR su$="!" OR su$="?" THEN RETURN
3050    NEXT i
3990 :
3992 RETURN
3995 :
4000 REM Satzzuweisung
4100   s$(s)=LEFT$(te$,i)
4140      PRINT
4150      PRINT "Satz Nr. ";s;" = ";s$(s)
4200   s=s+1
4300   te$=RIGHT$(te$,LEN(te$)-i)
4350        PRINT "Resttext..Nr.";s;"...";te$
4400   IF te$="" THEN 600
4500 RETURN
7000 :
10000 REM w.ten Satz in Woerter aufteilen
```

```
10030    se=1
10040    i=0
10050    wo$=s$(w)
10060     IF LEFT$(wo$,1)=" " THEN wo$=RIGHT$(wo$,LEN(wo$)-1)
10100     i=i+1
10130       sp$=MID$(wo$,i,1)
10150        IF sp$=" " THEN GOSUB 11000
10160        REM Leerzeichen gefunden
10200     IF i<=LEN(wo$) THEN 10100
10210   GOSUB 11000
10215 RETURN
10990 :
11000 REM Wort isolieren
11010    w$(w,se)=LEFT$(wo$,i)
11020    i=i-LEN(w$(w,se))
11030       se=se+1
11040       REM Satzelementnummer erhoehen
11100          wo$=RIGHT$(wo$,LEN(wo$)-LEN(w$(w,se-1)))
11110          REM Restsatz festlegen
11200 RETURN
11300 :
20000 REM grammatische Analyse
20005   PRINT "Grammatische Analyse des Satzes Nr.";w
20007   PRINT
20008   ORIGIN 225,237:DRAW 410,0
20010    REM Beseitigung von Leerzeichen und Satzzeichen
20015     af=0
20020     FOR x=1 TO 10
20030       y=w
20040          IF w$(y,x)<>"" THEN w$(y,x)=LEFT$(w$(y,x),
               LEN(w$(y,x))-1)
20060      NEXT x
20070 :
20100 REM Wort selektieren
20105   se=1
20110   wo$=w$(w,se)
20120     GOSUB 25000
20130     REM Vergleich
20140     se=se+1
20150    IF w$(w,se)="" THEN GOTO 30000
20160    REM Satz beenden
20170 GOTO 20110
24990 :
25000 REM Vergleich
25090 :
25100 REM Nomen?
25105    vg$="Nomen"
25110      FOR i=1 TO 100
25120        IF no$(i)=wo$ THEN GOTO 29000
25130        IF no$(i)="*" THEN 25200
25150      NEXT i
25190 :
25200 REM Verb?
25205    vg$="Verb"
25210      FOR i=1 TO 100
25220        IF ve$(i)=wo$ THEN GOTO 29000
```

```
25230       IF ve$(i)="*" THEN 25300
25250     NEXT i
25290 :
25300 REM Praeposition?
25305   vg$="Praeposition"
25310     FOR i=1 TO 100
25320       IF pr$(i)=wo$ THEN GOTO 29000
25330       IF pr$(i)="*" THEN 25400
25350     NEXT i
25390 :
25400 REM Artikel?
25405   vg$="Artikel"
25410     FOR i=1 TO 100
25420       IF ar$(i)=wo$ THEN GOTO 29000
25430       IF ar$(i)="*" THEN 25500
25450     NEXT i
25490 :
25500 REM Adjektiv?
25505   vg$="Adjektiv"
25510     FOR i=1 TO 100
25520       IF ad$(i)=wo$ THEN GOTO 29000
25530       IF ad$(i)="*" THEN. 25600
25550     NEXT i
25580 IF wo$=" " OR wo$="" THEN RETURN
25590 :
25600 PRINT"Das Wort ";wo$;" ist nicht bekannt!"
25605   nichtbek$(af)=wo$
25610   af=af+1
25620 RETURN
25630 :
29000 PRINT "Das Wort ";wo$;" ist ein ";vg$
29010 RETURN
29800 :
30000 REM bisher unbekanntes Vokabular festlegen
30010   IF af=0 THEN 30340
30012   REM nichts unbekannt!
30015 PRINT:PRINT"Druecken Sie eine Taste zum Vok.Festlegen"
30016 q$=INKEY$:IF q$="" THEN 30016
30020 :
30100 WINDOW 55,76,15,25
30110 PAPER 1:PEN 0
30120 CLS
30130 :
30150 PRINT"Legen  Sie  nun   den Satzteil fest __>"
30200   FOR i=0 TO af-1
30210    PRINT i+1;"__";nichtbek$(i);
30220     q$=INKEY$:IF q$="" THEN 30220
30240      IF q$="n" THEN no$(no)=nichtbek$(i):no=no+1:GOTO 30300
30250      IF q$="t" THEN ar$(ar)=nichtbek$(i):ar=ar+1:GOTO 30300
30255      IF q$="v" THEN ve$(ve)=nichtbek$(i):ve=ve+1:GOTO 30300
30260      IF q$="a" THEN ad$(ad)=nichtbek$(i):ad=ad+1:GOTO 30300
30270      IF q$="p" THEN pr$(pr)=nichtbek$(i):pr=pr+1:GOTO 30300
30280    GOTO 30220
30300    PRINT" __> ";q$;" OK!"
30330   NEXT i
30340 :
```

```
30350 PRINT:PRINT "Weiter durch Tasten-  druck..."
30360  q$=INKEY$
30370  IF q$="" THEN 30360
30380 :
30400  WINDOW 30,79,10,25
30410  PAPER 0:PEN 1
30500 RETURN
31000 :
40000 REM Ende eines Durchganges
40005  CLS
40010   PRINT"Ende des Parsingdurchganges."
40020   PRINT
40030    PRINT"Druecken Sie..."
40040    PRINT
40060     PRINT "S ____ Statusmeldung"
40070     PRINT "E ____ Parsing beenden"
40100       q$=INKEY$:IF q$="" THEN 40100
40200         IF q$="e" THEN PRINT:CLS:STOP
40220       IF q$<>"s" THEN 40100
40240 :
40300   CLS:PRINT"Statusmeldung ___>"
40310     PRINT
40400       PRINT no;" __ Nomen"
40410       PRINT ar;" __ Artikel"
40420       PRINT ve;" __ Verben"
40430       PRINT ad;" __ Adjektive"
40440       PRINT pr;" __ Praepositionen"
40460      PRINT
40470    PRINT"Druecken Sie eine Taste..."
40480   q$=INKEY$:IF q$="" THEN 40480
40490 GOTO 40000
```

2.3 Generieren intelligenter Antworten mit dem CPC

Nachdem wir uns mit den Grundprinzipien intelligenter Sprachdekodierung und mit dem Parsing-Prozeß befaßt haben, wenden wir uns nun der Generierung von Antworten des Computers auf Eingaben in natürlicher Sprache zu. Sie werden sehen, daß sich die im vorherigen Abschnitt vorgestellten Methoden zur Zerlegung der Eingabesätze und zur internen Informationsrepräsentation sowie die Dekodierung noch häufiger verwenden lassen.

Fangen wir vorerst einmal ganz bescheiden an. Was soll ein Programm, das zum Dialog fähig sein soll, eigentlich genau ausführen? Das Minimum, das wir von ihm verlangen können, ist doch, daß es von Ihnen vorgenommene Eingaben sinnvoll und richtig wiedergibt:

Auf den Satz	»Ich mag gern gute Musik«
erwartet man die Antwort	»Du magst gern gute Musik.«

Genauso sollte die Eingabe »Du magst die französische Küche.«
mit »Ich mag die französische Küche.«
beantwortet werden.

Wenn Sie sich die vier Sätze einmal anschauen, werden Sie feststellen, daß, um die geforderten Antworten zu geben, lediglich die Subjekte und die Prädikate der Eingabesätze geändert werden müssen. Der Rest der Sätze

»gern gute Musik«

und

»die französische Küche.«

werden unverändert in den Antwortsatz übernommen.

2.3.1 Das BASIC-RAC-Programm

Das entsprechende Programm »BASIC-RAC«, das die in unserem Beispiel geforderte Umwandlung der Sätze vornimmt, sieht folgendermaßen aus:

```
1 REM Kuenstliche Intelligenz auf dem CPC
2 REM Titel: Basis-Rac
3 REM Satzumwandlung - Antworten
5 :
10   MODE 1
20   PAPER 0:PEN 1:BORDER 9,9
30 :
100 PRINT "BASIS-RAC __ DEMO _____ Olaf Hartwig '86"
105 WINDOW 3,38,5,10
106 PAPER 2:PEN 0:CLS
107 :
110 PRINT:PRINT " Eingabesatz ___>":PRINT
120 INPUT " ";sa$
125 :
130 REM Dekodierung der Eingabe
140       su$=LEFT$(sa$,3)
150            REM Subjekt isolieren
160       sa$=RIGHT$(sa$,LEN(sa$)-3)
170 :
180         FOR i=2 TO 6
200            IF MID$(sa$,i,1)=" " THEN 240
220         NEXT i
230 :
240       ve$=LEFT$(sa$,i)
250           REM Verb bestimmen
260       sa$=RIGHT$(sa$,LEN(sa$)-i)
270           REM Restsatzteil festlegen
280 :
300 REM Konjugation
320        IF su$="Ich"     THEN su$="Du"
340        IF su$="Du "     THEN su$="Ich "
```

```
360        IF ve$=" mag "    THEN ve$="magst"
380        IF ve$="magst "   THEN ve$="mag "
400 :
500 REM Ausgabe
505   WINDOW 3,38,12,20
507    PAPER 1:CLS:PRINT
508     PRINT " Transformierte Ausgabe__>":PRINT
510     PRINT " ";su$;" ";ve$;" ";sa$
520 PRINT:STOP
```

Der Programmaufbau ist einfach. Aus dem in der Variablen »sa$« abgelegten Eingabesatz wird zunächst das Subjekt, das am Anfang des Satzes steht, isoliert und in der Variablen »su$« gespeichert. In Zeile 160 wird der Restsatz ohne dieses Subjekt bestimmt. Anschließend beginnt das Programm die Suche nach einem Leerzeichen, welches den zweiten Satzteil, das Prädikat, vom übrigen Teil des Satzes trennt. Das Prädikat wird in der Variablen »ve$« abgespeichert.
Dann wird der Satzrest, also der Teil des Satzes ohne Subjekt und Prädikat, in »sa$« isoliert. Das Programm beginnt daraufhin die Konjugation. Zunächst wird das Subjekt, dann das Prädikat geändert.
Die Bildschirmgestaltung des BASIC-RAC-Programms ist im folgenden dargestellt.

BASIC-RAC DEMO (Olaf Hartwig)

--> Eingabesatz:
? Ich mag gerne joggen!
--> Transformierte Ausgabe:
Du magst gerne joggen!

Das Programm funktioniert für die gewählten Beispiele, geben Sie aber beispielsweise einen Satz ein wie

»Ich spiele ausgezeichnet Tennis.«

so gibt es Ihnen die recht unintelligente Antwort

»Du spiele ausgezeichnet Tennis.«

Das Manko ist schnell behoben, indem die folgende Zeile in das Programm eingefügt wird:

```
390 if ve$="spiele" then ve$="spielst"
```

Und schon erhält man die richtige Antwort

»Du spielst ausgezeichnet Tennis.«

Um einen sinnvollen Dialog führen zu können, muß bedingt durch den Algorithmus des BASIC-RAC-Programms der Konjugationsteil durch die Berücksichtigung von so vielen Umwandlungsfällen wie möglich immer umfangreicher werden.
Jedes Prädikat und Subjekt will gespeichert sein. So kommen wir also nicht weiter, abgesehen vom zunehmenden Umfang der Wissensbasis ist die Satzabfolge zudem nicht immer regelmäßig, sie entspricht also nicht immer der Folge

Subjekt/Prädikat/Restsatz.

Vielmehr kann das Subjekt mitten im Satz stehen, es muß auch nicht immer ein Personalpronomen sein. Außerdem sollte ein Satz wie

»Dein Surfboard gefällt mir.«

sinnvollerweise folgendermaßen mit der Aussage des Dialogpartners, also in diesem Fall des Computers, übersetzt werden:

»Mein Surfboard gefällt dir.«

Damit das vorgestellte BASIC-RAC-Programm dies bewerkstelligt, muß seine Datenbank, in der Konjugationsregeln enthalten sind, sehr groß werden. Wir werden daher das Konzept im folgenden neu gestalten und das Programm MICRO-DBABY entwickeln, das mit einem geringeren Aufwand bessere Ergebnisse erzielt.
Sie können zuvor gerne mit dem bisher entwickelten Konzept weiter experimentieren. Wenn Sie das Programm immer mehr erweitern, können sich damit u.U. recht bemerkenswerte Gespräche führen lassen.

2.3.2 Der Original-RACTER

In den USA gibt es ein ebenfalls ganz in BASIC geschriebenes intelligentes Dialogprogramm namens RACTER (von franz. Raconteur, Geschichtenerzähler). Das Programm funktioniert in einigen Teilen nach den bereits vorgestellten Grundprinzipien.
Es ist ein perfektes Beispiel für »hausgemachte« Programme. Am Anfang existierte nur ein kurzes Programm wie beispielsweise das zuvor vorgestellte BASIC-RAC-Programm. Im Laufe der Zeit hat sich RACTER immer mehr »aufgebläht«, immer mehr und immer raffiniertere »Programm-Lagen« wurden um die älteren, einfacheren Routinen gewickelt.
Inzwischen besteht RACTER aus einem Software-Wust, den niemand je auseinandergenommen, analysiert, neu strukturiert und dokumentiert hat. Einige Sprachwissenschaftler vergleichen den Aufbau RACTERs mit der menschlichen Sprache. Auch diese besteht aus einer unhandlichen und verstrickten, sehr komplizierten Anhäufung vielfältiger Regeln.
Ich kann es mir hier nicht verkneifen, Ihnen einen Dialog zwischen dem ELIZA-Programm, das wir später noch gemeinsam entwickeln werden, und RACTER vorzustellen.
Ein kurzer Ausschnitt aus dem Dialog:

ELIZA: Hallo, hast du irgendwelche Probleme?
RACTER: Natürlich nicht! Sag, bist du arm?
ELIZA: Warum fragst du das?
RACTER: Ich frage, weil ROMs so sind, weißt du? Außerdem ist es vielleicht an der Zeit, die Frage zu stellen. Ich möchte aber, daß du meine Frage beantwortest. Bist du arm?
ELIZA: Warum interessiert dich das so sehr? Was hältst du von Geld?

RACTER: Ich finde es in Ordnung, Geld zu mögen. Nimmst du mir das übel? Ich weiß nicht, ob ich Geld mag. Ich frage mich, ob Almosenempfänger meine Einstellung zu Geld mögen. Ich bestehe aus Silizium. Was sind deine besten Eigenschaften?

.

.

.

Und so geht das ewig weiter. RACTER ergeht sich in weiten Abschweifungen und endlosen Assoziationsketten. Zu einem ähnlichen Programm könnten Sie gelangen, wenn Sie das Konversationsprogramm BASIC-RAC nach dem vorgestellten Verfahren immer mehr erweitern und es später mit einem Programm vom Typ ELIZA kombinieren. Grenzen werden hierbei nur durch den auf dem CPC verfügbaren Speicherplatz gesetzt. Die Funktionsweise von RACTER basiert darauf, daß zunächst einzelne Schlüsselwörter gesucht und so Assoziationen generiert werden.

2.3.3 Das MICRO-DBABY-Programm

Der vorgestellte Algorithmus des BASIC-RAC-Programms hat den großen Nachteil, daß er schnell zu einer nicht mehr überblickbaren Anhäufung von unterschiedlichen Konjugationsformen heranwächst. Es besteht aber die Möglichkeit, einen einfacheren und eleganteren Lösungsansatz zu realisieren.

Wie bereits versprochen, stelle ich Ihnen hier MICRO-DBABY vor. Dieser Dialogpartner ist nicht nur in der Lage, bei einem begrenzten Programmieraufwand in vielen Fällen korrekte und sinnvolle Antworten zu geben, sondern besitzt außerdem eine gewisse Lernfähigkeit.

Im Dialog mit Ihnen erweitert das Programm seinen Wortschatz und gibt neue oder leicht modifizierte Antworten, es wiederholt also nicht nur den zuvor eingegebenen Text. Anfangs besitzt MICRO-DBABY kein Grundvokabular zur Ermittlung der Wortarten. Implementiert sind lediglich ein Satzdekodierungsteil sowie ein Konjugationsteil.

Der Konjugationsteil ist so gestaltet worden, daß verstärkt allgemeine Regeln implementiert wurden, die einen Großteil der möglichen Eingabesätze korrekt umwandeln können. Schauen wir uns nun den Programmaufbau an:

Der MICRO-DBABY-Supervisor

Der Programmkopf des Programms MICRO-DBABY ruft sechs Module auf. Er lautet folgendermaßen:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: Micro-Dbaby Version 1
5 :
6  MODE 1
7  BORDER 17,17
8  WIDTH 60
10 :
```

```
30 PAPER 3
35 PEN 1
40 CLS
50 :
100 GOSUB 500
110     REM Initialisation
120 :
200     GOSUB 700
210        REM Eingabe
300         GOSUB 1000
310            REM Satzdekodierung
350         GOSUB 2500
360            REM Syntaxfestlegung
370         GOSUB 3500
380            REM Satzausgabe
400    GOTO 200
410       REM naechster Satz
420 :
450 GOSUB 4000
460    REM MICRO-Dbaby Ende
470 STOP
```

Der Programmkopf legt die Ausgangsbedingungen der Bildschirmgestaltung fest. In den Funktionsaufrufen werden zwei Routinen, die Initialisation sowie der Endteil, nur einmal aufgerufen, die anderen Routinen bilden eine Programmschleife. In dieser Schleife muß zunächst von Ihnen ein Satz eingegeben werden, der dann dekodiert und syntaktisch aufbereitet wird. Anschließend erfolgt die Textausgabe durch MICRO-DBABY. Der Vorgang wiederholt sich während des gesamten Dialogs.

Der Initialisationsteil

Hier werden die für das von MICRO-DBABY zu erlernende Vokabular benötigten Felder und Hilfsfelder dimensioniert. Ferner wird der Programmkopf angezeigt.
Der Teil lautet:

```
500 REM Initialisation
505     DIM w$(100)
508       REM Woerter des Eingabesatzes
510     DIM v$(100)
515       REM Vokabular
520     DIM sh$(100)
525       REM Satzhilfsvariable
530 :
550   CLS
560     PRINT "MICRO-DBABY ____ Dialog in nat. Sprache"
562       ORIGIN 0,380:DRAW 660,0
565       PRINT:PRINT TAB(14)"Copyright  Olaf Hartwig '85";
570       PRINT
580      WINDOW 4,40,4,25
585     PAPER 1
587     PEN 4
590    CLS
600 RETURN
```

Die Texteingabe

Der von Ihnen eingegebene Text innerhalb des Dialogs wird in die Satzvariable »s$« übernommen. Das geschieht über einen einfachen INPUT-Befehl. Wenn Sie wollen, können Sie die Eingabe auch durch die im DOC-Programm vorgestellte Eingaberoutine vornehmen lassen. So werden dann auch hier nicht erlaubte Zeichen wie Kommata angenommen.
Die Eingabeprozedur prüft, ob Sie den Dialog durch die Befehle »ende«, »schluss« oder »beenden« abbrechen wollen. Anschließend werden die Pointer für die Wortlänge (wl) sowie die Wortebene (we) gesetzt. Der Text wird in die Satzhilfsvariable »sh$« übernommen.
Die Realisierung in der Routine sieht wie folgt aus:

```
700 REM Eingabe
705 :
710 PRINT:PRINT"_>";:INPUT s$
712   REM  Satzeingabe
715    IF s$="stop" OR s$="ende" OR s$="schluss" OR s$="beenden"
       THEN 450
720     sh$=s$
730     s$=""
740    wl=1
750    we=1
760 :
770 RETURN
```

Das Kernprogramm

Es besteht aus zwei Teilen. In der ersten Subroutine wird der von Ihnen in der Variablen »sh$« abgelegte Text auf Leerzeichen und damit auf einzelne Wörter hin durchsucht. Ist ein Wort gefunden worden, so springt das Programm in die zweite Subroutine, die die interne Datenrepräsentation regelt.
Das letzte Wort eines Satzes wird gesondert gespeichert und auf Satzzeichen untersucht, die, falls vorhanden, gelöscht werden.
Die erste Subroutine:

```
1000 REM Initialisation
1050 :
1070   FOR i=1 TO LEN(sh$)
1080      IF MID$(sh$,i,1)=" " THEN GOSUB 2000
1090      REM Wort gefunden
1095 NEXT i
1097 :
1200 REM letztes Wort
1205 w$(we)=MID$(sh$,wl)
1220    p$=RIGHT$(w$(we),1)
1225    IF p$="." OR p$="!" OR p$="?" THEN w$(we)=LEFT$(w$(we),
        LEN(w$(we))-1)
1230 :
1240 RETURN
```

Die zweite Subroutine des Dekodierungs-Programms wird nach jedem von der vorherigen Routine gefundenen Wort aufgerufen. Hier werden die einzelnen Wörter im »w$()«-Feld abgelegt. Die Wortebene (we) sowie die Wortlänge (wl) werden neu festgelegt:

```
2000 REM Zuweisung w$()feld
2010   w$(we)=MID$(sh$,wl,i-wl)
2020   wl=i
2030   we=we+1
2040 RETURN
```

Die Syntaxfestlegung der Ausgabe

Dieser Programmteil legt das grammatische Format der Ausgabe sowie die Konjugationen fest und verändert somit die Syntax der Textausgabe. Er besteht ebenfalls aus zwei Unterprozeduren.
Die erste Prozedur übernimmt die Elemente des »w$()«-Feldes in das Vokabulararray »v$()«. Anschließend erfolgt in der zweiten Subroutine in den Zeilen 3000 ff. die Syntaxumwandlung. Nachdem der Schleifenvariablen »k« der Wert der Wortebene zugewiesen wurde, wird die Vokabularebene (e) um ein Element erhöht. Der eingegebene Satz wird im Satzfeld »sh$()« abgelegt:

```
2500 REM Syntaxfestlegung
2520    FOR k=1 TO we
2530       v$(k)=w$(k)
2540    NEXT k
2550       GOSUB 3000
2560       k=we
2580    sh$(e)=sh$
2600       e=e+1
2610       IF e>99 THEN e=20
2620    sh$=""
2650 RETURN
```

In der zweiten Subroutine erfolgt die eigentliche Umwandlung der Syntax. Alle Worte des Satzes werden in der »FOR I...NEXT I«-Schleife nacheinander analysiert.
Das sieht so aus:

```
3000 REM Umwandlung Syntax
3005 yp=VPOS(#0)
3010 PRINT
3012 :
3015  WINDOW 30,39,6,24:PAPER 2:PEN 0:CLS
3020  PRINT "DekodierteEingabe_>":PRINT
3030 FOR i=1 TO we
3040   z$=v$(i)
3041   IF i=1 THEN z$=" "+z$
3042   PRINT">";MID$(z$,2)
3045     IF z$="dich" OR z$=" dich" THEN w$(i)=" mich"
3046     IF z$="wiederhole" THEN 3300
3047     IF z$=" haelst" THEN w$(i)=" halte":z$=""
3048     IF z$="ist" OR z$=" ist" THEN 3300
3050     IF z$="war" OR z$=" war" THEN w$(i)=" ist gewesen":
         GOTO 3300
```

```
3052      IF z$=" mich" THEN w$(i)=" mir"
3055      IF z$="dir" OR z$=" dir" THEN w$(i)=" mir"
3056      IF z$="kannst" THEN w$(i)="kann":GOTO 3300
3057      IF z$="magst" THEN w$(i)="mag":GOTO 3300
3058      IF z$="warst" THEN w$(i)="war":GOTO 3300
3060      IF z$="ich" OR z$=" ich" THEN w$(i)=" du"
3062      IF z$="willst" THEN w$(i)=LEFT$(z$,4):z$=""
3063      IF RIGHT$(z$,2)<>"st" THEN 3067
3064      IF LEN(z$)<4 THEN 3067
3065         IF MID$(z$,LEN(z$)-2,1)="e" THEN w$(i)=LEFT$
             (z$,LEN(z$)-2):GOTO 3067
3066         w$(i)=LEFT$(z$,LEN(z$)-2)+"e"
3067      IF RIGHT$(z$,2)="le" AND LEN(z$)>4 THEN w$(i)=LEFT$
          (z$,LEN(z$)-2)+"st"
3070      IF z$="mir" OR z$=" mir" THEN w$(i)=" dir"
3075      IF z$=" bist" OR z$="bist" THEN w$(i)=" bin"
3080      IF z$=" ist gewesen" THEN w$(i)=" war"
3090      IF z$=" mein" THEN w$(i)=" dein"
3100      IF z$=" bin" THEN w$(i)=" bist"
3110      IF z$="du" OR z$=" du" THEN w$(i)=" ich"
3120      IF z$="du bist" OR z$=" du bist" THEN w$(i)=" ich bin"
3130      IF z$="dein" OR z$=" dein" THEN w$(i)=" mein"
3150      IF RIGHT$(z$,7)="ich bin" THEN w$(i)=LEFT$(z$,
          LEN(z$)-7)+"du bist"
3160      IF v$(z+1)=" sind" AND z$+v$(i+1)="ihr seid"
          THEN w$(i+1)="bin"
3170      IF z$="wir haben" THEN w$(i)=" ich habe"
3180      IF z$="wir wuerden" THEN w$(i)=" ich wuerde"
3190      IF RIGHT$(z$,4)=" der" THEN w$(i)=LEFT$(z$,LEN(z$)-4)
3210      IF RIGHT$(z$,4)=" ein" THEN w$(i)=v$(i+1)
3220      IF RIGHT$(z$,5)=" eine" THEN w$(i)=v$(i+1)
3230 :
3300 NEXT i
3310 :
3330 PRINT:PRINT"Ende."
3340 :
3350   WINDOW 4,29,4,25:PAPER 1:PEN 4
3360   LOCATE 1,yp
3370 :
3400 RETURN
```

Die implementierten Umwandlungsregeln bestehen aus zwei Grundtypen:
Zum einen werden einzelne Wörter wie in unserem vorher entwickelten BASIC-RAC-Programm direkt verglichen und umgewandelt. Das wird beispielsweise dazu verwendet, um Possessivpronomen und Personalpronomen umzuwandeln:
Das Possessivpronomen

»mir«

wird so zu

»dir«

geändert. Aus dem Personalpronomen

»ich«

entsteht

»du« usw.

Diese direkte Umwandlung wird außerdem dazu verwendet, um unregelmäßige Verbformen »in den Griff« zu bekommen.
So wird aus

»war«

das Verb

»ist gewesen« etc.

Die zweite zur Umwandlung verwendete Methode ist die Anwendung allgemeingültiger Regeln. So wandeln beispielsweise die Zeilen 3063 gekoppelt mit 3067 Verben nach folgender Regel um: Aus

»ich spiele«

wird

»du spielst«

und

»ich eile«

wird zu

»du eilst«

Enthält also ein Verb die letzten beiden Zeichen »le«, werden diese vom Verbstamm abgeschnitten und durch die Endung »st« ersetzt. Die entsprechende umgekehrte Regel, also die Umwandlung von »du spielst« in »ich spiele«, erfolgt in den Zeilen 3063, 3064, 3065 und 3066.
Bei den vielen Sonderfällen der deutschen Grammatik ist es leider unvermeidbar, daß einige Wörter durch das recht kurze CPC-Programm nicht ganz korrekt konjugiert werden.
(Anmerkung: Obwohl hier des besseren Verständnisses wegen jeweils das Pronomen in Verbindung mit dem Verb genannt wird, erfolgt die Umwandlung natürlich für jedes einzelne Wort getrennt.)

Der MICRO-DBABY-Antwortteil

Dieser Programmteil gibt die im Feld »w$()« in der syntaktisch richtigen Reihenfolge enthaltenen Wörter der Antwort aus. Er lautet:

```
3500 REM Antwort
3505  PRINT
3506   PEN 2
```

```
3507     PRINT "MDB_> ";
3508       PEN 0
3510           FOR i=1 TO k
3520                PRINT w$(i);
3530           NEXT i
3540     PRINT
3550  RETURN
```

Der Endteil

Die Prozedur wird direkt aus der Eingabeprozedur aufgerufen, wenn Sie die Wörter »schluss«, »ende« oder »beenden« eingeben. Sie gibt eine Abschlußbemerkung aus und springt anschließend zum STOP-Befehl in der Zeile 470:

```
4000 REM Ende
4020      PRINT
4030        PRINT
4040          PRINT "Ok, vielen Dank fuer das"
4050          PRINT "Gespraech! Tschuesss..."
4060      PRINT
4070 RETURN
```

Ein Demonstrationsdialog

Was leistet nun MICRO-DBABY? Schauen wir uns einmal einen Beispieldialog an:

ANWENDER : hallo wie geht es dir?

(Es folgt die Anzeige der zerlegten Wörter:)
>hallo
>wie
>geht
>es
>dir

Ende.

MDB.> : hallo wie geht es mir
ANWENDER : magst du gerne im meer schwimmen?
MDB.> : mag ich gerne im meer schwimmen
ANWENDER : dein sprachstil gefällt mir!
MDB.> : mein sprachstil gefällt dir
ANWENDER : möchtest du heute im stadion fussball spielen?
MDB.> : möchte ich heute im stadion fussball spielen
.
.
.

MDBABY gibt meistens recht korrekte Antworten. Es existieren aber u. a. einige grammatisch unregelmäßige Verbformen, die MDBABY etwas verfälscht umwandelt. Das kann behoben werden, indem man den Syntax-Umwandlungsteil für die jeweiligen Verbformen entsprechend erweitert.
Beachten Sie bitte, daß der Dialog in Kleinbuchstaben erfolgen sollte, da der Umwandlungsteil alle Wortmuster in Kleinschreibung beinhaltet.
Ich möchte Ihnen hier noch eine kleine Änderung mit recht großen Auswirkungen für den Dialog vorstellen. Geben Sie im Ausgabeteil die folgende Erweiterung ein:

```
3510 for i=1 to k+1
```

Was passiert?

Bei der Neubelegung der »W$()«-Felder werden die alten Vokabeln lediglich überschrieben, nicht aber gelöscht. Ist ein neuer Satz kürzer als der vorangegangene, so wird an MDBABYs Antwort noch jeweils ein weiteres Wort angehängt.
Ein Dialog kann nun wie folgt lauten:

ANWENDER : was hältst du von hibab?
MDBABY : was halte ich von hibab
ANWENDER : sag wie heisst du?
MDBABY : sag wie heisse ich hibab
ANWENDER : wiederhole das noch einmal!
MDBABY : wiederhole das noch einmal hibab
ANWENDER : du heisst wirklich hibab?
MDBABY : ich heisse wirklich hibab hibab
ANWENDER : ganz bestimmt?
MDBABY : ganz bestimmt wirklich
.
.
.

Experimentieren Sie ruhig selbst mit dem Programm. Sie können MDBABY beliebig verändern und erweitern und Ihren eigenen Wünschen anpassen.
Auf den folgenden Seiten sehen Sie der besseren Übersichtlichkeit halber das komplette Listing des MICRO-DBABY-Programms in der modifizierten Version 2:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: Micro-Dbaby Version 2
5 :
6  MODE 1
7  BORDER 17,17
8  WIDTH 60
20 :
30 PAPER 3
35 PEN 1
40 CLS
50 :
```

```
100 GOSUB 500
110     REM Initialisation
120 :
200     GOSUB 700
210        REM Eingabe
300         GOSUB 1000
310            REM Satzdekodierung
350         GOSUB 2500
360            REM Syntaxfestlegung
370         GOSUB 3500
380            REM Satzausgabe
400     GOTO 200
410       REM naechster Satz
420 :
450 GOSUB 4000
460    REM MICRO-Dbaby Ende
470 STOP
490 :
500 REM Initialisation
505     DIM w$(100)
508       REM Woerter des Eingabesatzes
510     DIM v$(100)
515       REM Vokabular
520     DIM sh$(100)
525       REM Satzhilfsvariable
530 :
550   CLS
560     PRINT "MICRO-DBABY II ____ intelligenter Dialog";
562       ORIGIN 0,380:DRAW 660,0
565       PRINT:PRINT TAB(14)"Copyright  Olaf Hartwig '85";
570       PRINT
580      WINDOW 4,40,4,25
585     PAPER 1
587     PEN 4
590    CLS
600 RETURN
610 :
700 REM Eingabe
705 :
710 PRINT:PRINT"_>";:INPUT s$
712   REM  satzeingabe
715    IF s$="stop" OR s$="ende" OR s$="schluss" OR s$="beenden"
       THEN 450
720     sh$=s$
730     s$=""
740    wl=1
750    we=1
760 :
770 RETURN
980 :
1000 REM Initialisation
1050 :
1070   FOR i=1 TO LEN(sh$)
1080      IF MID$(sh$,i,1)=" " THEN GOSUB 2000
1090      REM Wort gefunden
1095 NEXT i
```

```
1097 :
1200 REM letztes Wort
1205 w$(we)=MID$(sh$,wl)
1220    p$=RIGHT$(w$(we),1)
1225    IF p$="." OR p$="!" OR p$="?" THEN w$(we)=LEFT$(w$(we),
        LEN(w$(we))-1)
1230 :
1240 RETURN
1360 :
2000 REM Zuweisung w$()feld
2010   w$(we)=MID$(sh$,wl,i-wl)
2020   wl=i
2030   we=we+1
2040 RETURN
2060 :
2500 REM Syntaxfestlegung
2520    FOR k=1 TO we
2530       v$(k)=w$(k)
2540    NEXT k
2550       GOSUB 3000
2560       k=we
2580    sh$(e)=sh$
2600       e=e+1
2610       IF e>99 THEN e=20
2620    sh$=""
2650 RETURN
2700 :
3000 REM Umwandlung Syntax
3005 yp=VPOS(#0)
3010 PRINT
3012 :
3015  WINDOW 30,39,6,24:PAPER 2:PEN 0:CLS
3020  PRINT "DekodierteEingabe_>":PRINT
3030 FOR i=1 TO we
3040   z$=v$(i)
3041   IF i=1 THEN z$=" "+z$
3042   PRINT">";MID$(z$,2)
3045     IF z$="dich" OR z$=" dich" THEN w$(i)=" mich"
3046     IF z$="wiederhole" THEN 3300
3047     IF z$=" haelst" THEN w$(i)=" halte":z$=""
3048     IF z$="ist" OR z$=" ist" THEN 3300
3050     IF z$="war" OR z$=" war" THEN w$(i)=" ist gewesen":
         GOTO 3300
3052     IF z$=" mich" THEN w$(i)=" mir"
3055     IF z$="dir" OR z$=" dir" THEN w$(i)=" mir"
3056     IF z$="kannst" THEN w$(i)="kann":GOTO 3300
3057     IF z$="magst" THEN w$(i)="mag":GOTO 3300
3058     IF z$="warst" THEN w$(i)="war":GOTO 3300
3060     IF z$="ich" OR z$=" ich" THEN w$(i)=" du"
3062     IF z$="willst" THEN w$(i)=LEFT$(z$,4):z$=""
3063     IF RIGHT$(z$,2)<>"st" THEN 3067
3064     IF LEN(z$)<4 THEN 3067
3065         IF MID$(z$,LEN(z$)-2,1)="e" THEN w$(i)=LEFT$
             (z$,LEN(z$)-2):GOTO 3067
3066         w$(i)=LEFT$(z$,LEN(z$)-2)+"e"
```

```
3067      IF RIGHT$(z$,2)="le" AND LEN(z$)>4 THEN w$(i)=LEFT$
          (z$,LEN(z$)-2)+"st"
3070      IF z$="mir" OR z$=" mir" THEN w$(i)=" dir"
3075      IF z$=" bist" OR z$="bist" THEN w$(i)=" bin"
3080      IF z$=" ist gewesen" THEN w$(i)=" war"
3090      IF z$=" mein" THEN w$(i)=" dein"
3100      IF z$=" bin" THEN w$(i)=" bist"
3110      IF z$="du" OR z$=" du" THEN w$(i)=" ich"
3120      IF z$="du bist" OR z$=" du bist" THEN w$(i)=" ich bin"
3130      IF z$="dein" OR z$=" dein" THEN w$(i)=" mein"
3150      IF RIGHT$(z$,7)="ich bin" THEN w$(i)=LEFT$(z$,
          LEN(z$)-7)+"du bist"
3160      IF v$(z+1)=" sind" AND z$+v$(i+1)="ihr seid"
          THEN w$(i+1)="bin"
3170      IF z$="wir haben" THEN w$(i)=" ich habe"
3180      IF z$="wir wuerden" THEN w$(i)=" ich wuerde"
3190      IF RIGHT$(z$,4)=" der" THEN w$(i)=LEFT$(z$,LEN(z$)-4)
3210      IF RIGHT$(z$,4)=" ein" THEN w$(i)=v$(i+1)
3220      IF RIGHT$(z$,5)=" eine" THEN w$(i)=v$(i+1)
3230 :
3300 NEXT i
3320 :
3330 PRINT:PRINT"Ende."
3340 :
3350   WINDOW 4,29,4,25:PAPER 1:PEN 4
3360   LOCATE 1,yp
3370 :
3400 RETURN
3420 :
3500 REM Antwort
3505  PRINT
3506   PEN 2
3507    PRINT "MDB_> ";
3508      PEN 0
3510          FOR i=1 TO k+1
3520               PRINT w$(i);
3530          NEXT i
3540    PRINT
3550 RETURN
3570 :
4000 REM Ende
4020     PRINT
4030       PRINT
4040         PRINT "Ok, vielen Dank fuer das"
4050         PRINT "Gespraech! Tschuesss..."
4060     PRINT
4070 RETURN
```

2.4 Der intelligente Dialog

Bisher haben wir mit verschiedenen Verfahren zur Dekodierung von Texten in natürliche Sprache experimentiert, die interne Repräsentation von Daten im Speicher geklärt, und Verfahren zur syntaktischen Textanalyse kennengelernt.
Anhand des MICRO-DBABY-Programms sowie unseres BASIC-RAC-Programms, das das Grundprinzip des anschließend vorgestellten RACTER-Programms aufzeigte, wurden unterschiedliche Wege zur Generierung sinnvoller Antworten erprobt.
Der nächste Schritt führt zum intelligenten Dialog. Aufbauend auf einigen der bereits beschriebenen Algorithmen und mit weiteren, neuen KI-Techniken wird in diesem Abschnitt ein aus Modulen aufgebautes Programm entwickelt, das fähig ist, den in natürlicher Sprache eingegebenen Text innerhalb gewisser Grenzen zu verstehen.
Das Programm ist in der Lage, den Kontext, also den Sinn Ihrer in natürlicher Sprache eingegebenen Texte, in gewisser Weise zu erfassen. Anschließend generiert es sinnvolle, d. h. auf den Inhalt der Eingabe bezogene Antworten.
Wie zu Beginn des Buches erwähnt, beschränkt sich die Kommunikationsfähigkeit natürlichsprachiger Systeme fast ausschließlich auf eng begrenzte Wissensbereiche. Hier werden wir ein Programm vom Typ ELIZA entwickeln. Vielen von Ihnen wird der Name ELIZA bekannt sein. ELIZA ist in den letzten Jahren sicher mit eines der international bekanntesten Programme geworden.
Entwickelt wurde dieser Computer-Psychiater schon 1966 von dem MIT-Computer-Wissenschaftler Joseph Weizenbaum. ELIZA nimmt von seinem Benutzer natürlichsprachige englische Sätze an. Je nach Sinn bzw. Inhalt des Eingabetextes gibt ELIZA kontextbezogene Antworten.
ELIZA besteht aus zwei Programm-Modulen. Das eine Modul organisiert die Dekodierung der Sätze sowie die Ausgabe. Das zweite Modul enthält das Script. Weizenbaum verstand unter einem Script den Teil des Programms, der das Vokabular und einen »Regelsatz« enthält.
Je nach Inhalt des Scripts ist ELIZA in der Lage, sich mit dem Anwender über bestimmte Themengebiete zu unterhalten. ELIZA besitzt durch den modularen Aufbau eine gewisse Flexibilität. Am bekanntesten wurde das ELIZA-Programm mit dem Script eines Psychologen. Daher wollen wir uns jetzt auch näher mit dieser Version des ELIZA-Programms beschäftigen.

2.4.1 ELIZA als Psychotherapeut

Der bekannte Psychologe Carl Rogers ist der Ansicht, daß Psychotherapie am effektivsten ist, wenn sie ungerichtet abläuft, d.h. der Patient soll die Führung des Gesprächs übernehmen. Der Psychologe ermuntert ihn lediglich, über bestimmte Bereiche Näheres auszuführen.
Die Aufgabe erschien Weizenbaum für die Übertragung auf den Computer geeignet zu sein, und er entwickelte das Psychologenscript zu ELIZA.

Die Struktur von ELIZA

Beginnen wir mit dem Erstellen des deutschsprachigen Programms. Hier folgt zunächst wieder der Supervisor.

Er besteht aus sechs Prozeduren, von denen die ersten beiden, die Initialisierung und die Titelerzeugung, nur beim Programmstart aufgerufen werden. Die übrigen Unterprogramme sind die Satzeingabe, die Suchroutine, der Konjugationsteil sowie das Antwortsatzmodul. Sie bilden eine Schleife, in der der eigentliche Dialog abläuft. Der Programmkopf lautet:

```
3 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel :  Eliza Psychiater
5 :
6 MODE 2
7 WIDTH 60
10 :
30 PAPER 3
40 PEN 0
50 CLS
60 :
100 GOSUB 30000
110      REM Initialisation
150 GOSUB 20000
160      REM Titel
170 :
200      GOSUB 1000
210       REM Dialog Satzeingabe
300      GOSUB 2000
310       REM Suchroutine
400      GOSUB 3000
410       REM Konjugation
500      GOSUB 4000
510       REM Antwortsatz
610 GOTO 200
989 STOP
```

Der Initialisierungsteil

In dieser Routine wird das Script eines Psychologen in Datenfelder gelesen. Zunächst werden vier Felder für die beiden Code-Zahlen, das Schlüsselwortfeld sowie das Array für die Antworten dimensioniert:

```
30000 REM Initialisation
30010     DIM r(38): REM Codezahl 1
30020     DIM n(38): REM Codezahl 2
30030     DIM k$(38):REM Keywoerter
30040     DIM a$(118):REM Antworten
```

Anschließend werden zunächst Code-Zahlen in zwei Arrays gelesen. Worum geht es dabei? ELIZA sucht den eingegebenen Text beim Dekodieren nach bekannten Schlüsselwörtern ab. Wie das genau abläuft, werden Sie weiter unten näher kennenlernen.

Angenommen, ELIZA findet in einem Satz das Schlüsselwort Nummer drei. Die Schlüsselwörter werden später im Programm in ein Feld eingelesen. Das dritte Schlüsselwort lautet

»bist du«

Entdeckt ELIZA das aus zwei Wörtern bestehende Schlüsselwort, so schaut es in den beiden Code-Zahlen-Arrays nach, an welcher Stelle die zu dem Schlüsselwort passenden Antworten oder Kommentare stehen.
Wir werden nachher noch insgesamt 118 Antworten bzw. Bemerkungen in ein Datenfeld einlesen. Der erste Code-Zahlenwert des Feldes »r(3)« ist in dem Beispiel gleich sechs, d.h. die erste zu dem Schlüsselwort gehörende Bemerkung steht im Antwortfeld an sechster Stelle. Der Code-Wert »r(i)« stellt also die Anfangsstellung zu einem Schlüsselwort passender Kommentare innerhalb des Antwortenfelds dar.
Entsprechend repräsentiert der zweite Code-Zahlenwert des Feldes »n(i)« die Stellung des letzten passenden Kommentars in diesem Feld.
Das sieht in dem Programm folgendermaßen aus:

```
31000 REM Codezahlen lesen
31005    FOR i=1 TO 38
31010        READ r(i)
31020        READ n(i)
31050    NEXT i
31060 :
31100 REM data r(i),  n(i)
31110 DATA       1  , 3
31120 DATA       4  , 5
31130 DATA       6  , 9
31140 DATA       6  , 9
31150 DATA       10 , 13
31160 DATA       14 , 16
31170 DATA       17 , 19
31180 DATA       20 , 21
31190 DATA       22 , 24
31200 DATA       25 , 27
31210 DATA       28 , 31
31220 DATA       28 , 31
31230 DATA       32 , 34
31240 DATA       35 , 39
31250 DATA       40 , 48
31260 DATA       40 , 48
31270 DATA       40 , 48
31280 DATA       40 , 48
31290 DATA       40 , 48
31300 DATA       40 , 48
31310 DATA       49 , 50
31320 DATA       51 , 54
31330 DATA       55 , 58
31340 DATA       59 , 62
31350 DATA       63 , 63
31360 DATA       63 , 63
31370 DATA       64 , 68
31380 DATA       69 , 73
```

```
31390 DATA        74 , 75
31400 DATA        76 , 79
31410 DATA        80 , 82
31420 DATA        83 , 89
31430 DATA        90 , 92
31440 DATA        93 , 98
31450 DATA        99 , 105
31460 DATA        116, 118
31470 DATA        113, 115
31480 DATA        106, 112
```

Nachdem das Einlesen der Code-Zahlen erfolgt ist, müssen die oben bereits erwähnten Schlüsselwörter, die Key-Wörter, in das Feld »k$(i)« eingelesen werden. Der später im Dialog mit ELIZA eingegebene Text wird auf die folgenden Key-Wörter hin untersucht.
Der dies bewerkstelligende Programmteil lautet:

```
32000 REM Keywoerter lesen
32010    FOR i=1 TO 38
32020       READ k$(i)
32030    NEXT i
32070 :
32100 REM data Keywoerter
32110 DATA         kannst du
32120 DATA         kann ich
32130 DATA         bist du
32140 DATA         du bist
32150 DATA         ich kann
32160 DATA         ich fuehle
32170 DATA         kannst du
32180 DATA         kann ich
32190 DATA         bist du
32200 DATA         ich kann
32210 DATA         ich bin
32220 DATA         im
32230 DATA         du
32240 DATA         ich will
32250 DATA         was
32260 DATA         wie
32270 DATA         wer
32280 DATA         wo
32290 DATA         wann
32300 DATA         warum
32310 DATA         name
32320 DATA         grund
32330 DATA         leid
32340 DATA         traum
32350 DATA         hallo
32360 DATA         tag
32370 DATA         mag sein
32380 DATA         nein
32390 DATA         dein
32400 DATA         immer
32410 DATA         denke
32420 DATA         gleich
```

```
32430 DATA        ja
32440 DATA        freund
32450 DATA        computer
32460 DATA        eliza
32470 DATA        geld
32480 DATA        kein keywort
```

Alles, was noch für ein fertiges Script benötigt wird, sind die zu den jeweiligen Schlüsselwörtern gehörenden Antworten und Bemerkungen. Die insgesamt 118 Kommentare und Sätze werden im folgenden in das Antwortfeld »a$(i)« eingelesen:

```
33000 REM Antworten lesen
33010    FOR i=1 TO 118
33020        READ a$(i)
33030    NEXT i
33080 :
33100 REM Data antworten
33110 DATA glaubst du nicht daran dass ich das kann und zwar*
33120 DATA vielleicht wuerdest du das gerne tun naemlich*
33130 DATA du haettest es gerne wenn ich das koennte also
33140 DATA vielleicht moechtest du auch nicht*
33150 DATA moechtest du gerne*
33160 DATA warum glaubst du das ich das bin also*
33170 DATA gefaellt dir der gedanke dass ich so bin?
33180 DATA vielleicht wuerdest du gerne so sein also*
33190 DATA wuenscht du dir manchmal du waerst*
33200 DATA bist du nicht vielleicht*
33210 DATA warum bist du nicht*
33220 DATA moechtest du gerne dazu faehig sein das zu tun also*
33230 DATA beunruhigt dich das sehr stark oder weniger?
33240 DATA erzaehle mir doch etwas mehr ueber solche gefuehle.
33250 DATA fuehlst du dich oft*
33260 DATA geniest du dich so zu fuehlen?
33270 DATA glaubst du vielleicht ich bin nicht oder kann nicht*
33280 DATA vielleicht in besseren zeiten werde ich*
33290 DATA moechtest du dass ich*
33300 DATA glaubst du ich sollte faehig sein zu*
33310 DATA warum kannst du nicht*
33320 DATA warum bist du so daran interessiert ob oder ob
           ich nicht*
33330 DATA wuerdest du es vorziehen wenn ich nicht*
33340 DATA vielleicht bin ich in deiner phantasie*
33350 DATA woher weisst du dass ich das nicht kann?
33360 DATA hast du das schon versucht?
33370 DATA vielleicht kannst du nun*
33380 DATA kamst du zu mir weil du so bist also*
33390 DATA wie lange bist du schon*
33400 DATA glaubst du es ist normal so zu sein naemlich*
33410 DATA magst du das?
33420 DATA ich dachte wir spraechen ueber dich nicht ueber mich.
33430 DATA oh ich*
33440 DATA du sprichst da wirklich ueber mich nicht wahr?
33450 DATA was wuerde es fuer dich bedeuten wenn du das
           bekommen koenntest?
```

```
33460 DATA warum moechtest du*
33470 DATA angenommen du erhaelst bald*
33480 DATA was wuerde es fuer dich bedeuten wenn du das bekaemst?
33490 DATA ich wensche mir auch manchmal*
33500 DATA warum fragst du das?
33510 DATA interessiert dich diese frage sehr stark?
33520 DATA welche antwort gefiele dir am meisten?
33530 DATA was glaubst du?
33540 DATA kommen dir solche fragen oefter in den sinn?
33550 DATA was willst du eigentlich wissen?
33560 DATA hast du das schon andere personen gefragt?
33570 DATA hast du solche fragen schon mal zuvor gestellt?
33580 DATA was kommt dir sonst noch in den sinn wenn du
           das fragst?
33590 DATA namen interessieren mich weniger.
33600 DATA ich kuemmere mich nicht um namen...bitte fahre fort...
33610 DATA ist das der wirkliche grund oder verschweigst
           du mir etwas?
33620 DATA gaebe es vielleicht andere gruende-denk mal kurz nach...
33630 DATA erklaert dieser grund unter umstaenden noch
           etwas anderes?
33640 DATA welche andere gruende koennte es geben?
33650 DATA bitte entschuldige dich nicht!
33660 DATA entschuldigen sind nicht noetig...wirklich nicht!
33670 DATA wie fuehlst du dich wenn du dich entschuldigst?
33680 DATA sei doch nicht so defensiv!
33690 DATA was verraet dir dieser traum...was bedeutet er dir?
33700 DATA traeumst du eigentlich oft oder kommt das selten vor?
33710 DATA welche personen und welche gegenstaende erscheinen
           in deinen traeumen
33720 DATA beunruhigen dich solche traeume?
33730 DATA hallo wie geht es dir...erzaehle mir dein problem.
33740 DATA du bist dir da scheinbar nicht ganz sicher...gibt
           es dafuer gruende?
33750 DATA warum dieser unsichere ton?
33760 DATA kannst du nicht positiver denken?
33770 DATA du bist dir wirklich nicht sicher?
33780 DATA weist du nicht warum?
33790 DATA warum keine*
33800 DATA sag doch nicht immer nein...es ist so negativ.
33810 DATA warum nicht?
33812 DATA bist du dir sicher?
33814 DATA wirklich nicht?
33820 DATA warum bist du besorgt ueber mein*
33830 DATA wie steht es eigentlich mit deinem*
33840 DATA kannst du dabei an ein spezielles beispiel denken?
33850 DATA wann?
33860 DATA woran denkst du?
33870 DATA wirklich immer?
33880 DATA denkst du wirklich so?
33890 DATA aber du bist dir nicht sicher dass du*
33900 DATA du bezweifelst dass du*
33910 DATA in welcher form?
33920 DATA welche auswirkungen siehst du?
33930 DATA was zeigt dir dieser zusammenhang?
33940 DATA welche anderen verbindungen siehst du?
```

```
33950 DATA koennten da wirklich gemeinsamkeiten bestehen?
33960 DATA wie?
33970 DATA du scheinst dir ja ganz sicher zu sein!
33980 DATA bist du dir da auch sicher?
33990 DATA ich kann das nachvollziehen.
34000 DATA ich verstehe.
34010 DATA warum erwaehnst du freunde?
34020 DATA beunruhigen dich deine freunde?
34030 DATA aergern dich deine freunde?
34040 DATA bist du dir sicher dass du ueberhaupt freunde besitzt?
34050 DATA vielleicht liegt der grund deiner probleme bei
           deinen freunden?
34060 DATA vielleicht haben deine freunde negative
           auswirkungen auf dich.
34070 DATA beunruhigen dich computer?
34080 DATA sprichst du dabei eigentlich von mir im besonderen?
34090 DATA hast du angst vor maschinen?
34100 DATA hast du da gerade computer erwaehnt?
34110 DATA haben maschinen mit deinem problem zu tun?
34120 DATA glaubst du nicht dass computer den menschen
           helfen koennen
34130 DATA was an den maschinen stoert dich im besonderen?
34140 DATA sag hast du irgendwelche psychologischen probleme?
34150 DATA was sagt dir das? hat das eine bedeutung fuer dich?
34160 DATA aha! fahre bitte fort...
34170 DATA ich bin mir da nicht so ganz sicher ob ich dir
           da folgen kann...
34180 DATA kannst du deine gedanken etwas weiter ausfuehern?
34190 DATA wuerdest du das vielleicht etwas naeher beschreiben?
34200 DATA das ist ja interessant!
34210 DATA warum hast du probleme mit deinen mitmenschen?
34220 DATA glaubst du eigentlich dass geld alles bedeutet?
34230 DATA bist du dir sicher dass geld dein problem ist?
34240 DATA ich dachte wir sprachen ueber dich...nicht ueber mich!
34250 DATA was soll mit mir sein?
34260 DATA warum erwaehnst du immer meinen namen?
34270 :
36000 RETURN
```

Die Titelroutine

Mit den vorangegangenen Routinen ist der Scriptteil eines Psychologen fast vollständig. In den folgenden Abschnitten wird das bei allen ELIZA-Programmen annähernd gleichbleibende Verarbeitungsmodul vorgestellt.

Doch zunächst benötigen wir eine Titelroutine. Ihre Aufgabe ist es, den Programmtitel auszugeben und ein Textfenster sowie dessen Farbe festzulegen. Das sieht im Programm wie folgt aus:

```
20000 REM Titel
20010   PRINT
20020   PRINT TAB(12)" E L I Z A  ______  Psychiater"
20030   PRINT
20040   PRINT TAB(17)"(c) Olaf Hartwig  1985"
20050   PRINT
```

```
20060    WINDOW 6,79,6,25
20070    PAPER 2
20075    PEN 3
20080    CLS
20100 RETURN
```

Unser erstes Modul, der Psychologen-Scriptteil, ist damit schon komplett. Das Scriptmodul ist beliebig austauschbar. Sie können hier das Script eines Verkäufers, eines Programmierers, eines Politikers, Ihrer Freundin oder Ihres Freundes usw. einsetzen. In Anlehnung an den prinzipiellen Aufbau von ELIZA können Sie beliebig weiter experimentieren.
Vorerst wenden wir uns nun dem zweiten, in allen ELIZA-Programmen konstant bleibenden Steuerungsmodul zu:

Das ELIZA-Steuerungsmodul: Die Satzeingabe

Der von Ihnen während des Dialogs eingegebene Satz wird der Variablen „i$" zugewiesen. Ungültige Eingaben, d.h. eine leere Eingabe oder ein Text mit einem oder mehr als 200 Zeichen werden ignoriert.
Wenn später im Dialog die Anweisung »stop« eingegeben wird, wird das Gespräch beendet. Das sieht wie folgt aus:

```
1000 REM Dialog Satzeingabe
1100  PRINT
1110   INPUT "___> ";i$
1120    IF i$="" THEN 1110
1125    IF i$="stop" THEN STOP
1130    IF LEN(i$)<2 OR LEN(i$) >200 THEN PRINT
        "Ungueltige Eingabe...":GOTO 1110
1800 RETURN
```

Die Suchroutine

Diese Prozedur besteht aus zwei Modulen. Das erste Modul wählt je eins der 38 Schlüsselwörter aus und springt dann in das zweite Modul, welches den von Ihnen bereits eingegebenen Text nach diesen Key-Wörtern absucht:

```
2000 REM Suchschleife
2010 :
2020 REM Schluesselwort waehlen
2090 ma=0
2100   FOR i=1 TO 38
2110      zk$=k$(i)
2120        GOSUB 2500
2125          IF ma=9 THEN 2150
2130   NEXT i
2140 :
2150 RETURN
```

Wurde ein Schlüsselwort entdeckt, so setzt das zweite Modul den Marker »ma« auf den Wert 9. Dies ist für das Programm ein Zeichen, um die Suche abzubrechen und in das Hauptprogramm zu springen.
Das zweite Modul »scannt«, d.h. verschiebt und vergleicht den Text des Key-Wortes über den Eingabesatz. Das Verfahren funktioniert folgendermaßen:
Bisher haben wir beim Dekodieren von Texten jeweils die einzelnen Wortmuster direkt verglichen. Lautet beispielsweise das erste Wort eines Satzes »ich«, so verglich unser BASIC-RAC-Programm dieses Wort mit allen im Konjugationsteil enthaltenen Wörtern und kam schließlich zu der folgenden Programmzeile:

```
320 if su$="ich" then su$="du"
```

Die einzelnen Wörter werden also direkt verglichen und damit erkannt.
Im MICRO-DBABY-Programm haben wir eine weitere Methode zum Erkennen einzelner Wortmuster mit »Umwandlungsregeln« eingesetzt. Dabei erkannte das Programm in seinem Syntax-Umwandlungsteil Teile von Wörtern und wendete eine entsprechende Regel an. In der Programmdokumentation wurde beispielsweise die Umwandlung von der Verb-Endung »st« in »le« und umgekehrt erläutert.
Das in ELIZA angewendete Verfahren beschreitet einen anderen Teil zum Erkennen einzelner Wortmuster. Der auf ein Key-Wort zu untersuchende Text wird als Einheit angesehen und nicht mehr wie bisher in seine einzelnen Satzteile oder Wortarten zerlegt. Vielmehr werden die Satzteile jeweils einzeln mit denen des Schlüsselwortes verglichen.
Das Verfahren läßt sich an einem Beispiel veranschaulichen. Angenommen, das aktuelle Schlüsselwort lautet

»freund«

Der zu untersuchende Satz lautet

»mein freund kann ski fahren.«

Das Schlüsselwort wird nun am Satz entlang-»gescannt«:

Zielsatz:	»mein freund kann ski fahren.«
Key-Wort:	»freund«
Scanning um ein Zeichen nach rechts	freund --->

Der Vorgang erfolgt so lange, bis das Keywort »freund« im Satz entdeckt wurde. Im Computer sieht dies folgendermaßen aus:

Mit Schlüsselwort verglichener Textausschnitt:

Durchgang Nr.:
1. mein f
2. ein fr
3. in fre
4. n freu
5. freun
6. freund

Nach sechs Durchgängen ist der Textausschnitt identisch mit dem verglichenen Schlüsselwort und ELIZA hat das Wort identifiziert. Die neue Suchmethode hat große Vorteile gegenüber den bisher erörterten Verfahren. Angenommen, Sie geben in Ihrem Dialog den Satz

»meine freundin sieht fantastisch aus.«

ein. Auch in diesem Satz ist der Textausschnitt »freund« in dem Wort »freundin« enthalten und wird von ELIZA korrekt erkannt.
Ähnlich wird die Aussage

»ich habe viele freunde.«

behandelt. Die Suchmethode ist also wesentlich flexibler und universeller anwendbar.
Sie hat aber auch deutliche Nachteile. Im Satz

»heute ist freundliches wetter.«

findet ELIZA beispielsweise ebenfalls das Schlüsselwort »freund« in dem Wort »freundlich« wieder. Es besteht also die Gefahr von Doppeldeutigkeiten, die wir bei unseren bisherigen Suchmethoden in diesem Ausmaß nicht kannten.
Ein weiterer Nachteil des Verfahrens besteht darin, daß es komplexer als die Wortvergleichsmethoden ist, d.h. auch mehr Rechenzeit beansprucht. Das macht sich vor allem in BASIC mit seiner recht langsamen Ausführungsgeschwindigkeit störend bemerkbar. Sie müssen oft einige Sekunden auf die Antwort von ELIZA warten.
Welches Verfahren zur Erkennung von Wörtern jeweils angewendet werden sollte, hängt letztendlich von der Aufgabenstellung eines natürlichsprachigen Systems ab. Oft werden alle hier beschriebenen Algorithmen in umfangreichen KI-Systemen miteinander gekoppelt.
Unten sehen Sie das zweite Modul, das den eben erörterten Suchalgorithmus beinhaltet:

```
2500 REM Scanning
2510   FOR p=1 TO LEN(i$)
2520      w$=MID$(i$,p,LEN(zk$))
2530        IF w$=k$(i) THEN ma=9:k=i:RETURN
2540   NEXT p
2550   k=38:REM kein Keywort
2560 RETURN
```

Wurde das Schlüsselwort in der Zeile 2530 gefunden, so wird der Marker »ma« auf den Wert 9 gesetzt, was im ersten Modul einen Rücksprung in den Programmkopf bewirkt. Gleichzeitig wird der Key-Wort-Pointer »k« mit dem Wert der »FOR I–NEXT I«-Schleife gleichgesetzt. Er gibt die Stellung des Schlüsselwortes im Key-Wortfeld »k$()« an.
Sind mehrere Schlüsselwörter in Ihrem eingegebenen Satz enthalten, so wählt der Algorithmus immer das in seinem Key-Wortfeld weiter unten stehende Schlüsselwort aus. Das Vokabular des »k$()«-Feldes ist nach einer Prioritätsskala geordnet. Am Anfang des Feldes stehen häufig vorkommende Vokabeln wie

»ich«
»ich bin«
»ich fühle«

usw. Weiter unten im Feld befinden sich speziellere Schlüsselwörter wie

»freund«
»traum«
»geld«

oder

»computer«

Es ist natürlich für einen Psychologen sinnvoller, bei den selteneren Schlüsselwörtern nachzufragen und spezielle Fragen zu diesen Bereichen zu stellen.

Die Konjugation der Antworten ELIZAs

Die Konjugation untergliedert sich in die folgenden Programmteile:

Die Bestimmung des Antwortsatzes

Nachdem das Schlüsselwort höchster Priorität bestimmt worden ist, sucht ELIZA eine dazu passende Bemerkung oder Frage aus seinem Script aus:

```
3000 REM Konjugation
3010 REM Antwortsatz vorwaehlen
3020      zw=n(k)-r(k)
3030      zr=INT(RND(1)*zw)
3040    aw=r(k)+zr
3045    IF aw=aalt THEN 3000
3047    aalt=aw
3050 :
3060 mk=0
3070 REM Restsatzmarker reset
```

Die Variable »k« beinhaltet die Stellung des Schlüsselwortes im Key-Wortfeld. Die Variable »n(k)« stellt die Position des ersten Kommentars im Antwortfeld »a$()« dar, das Feld »r(k)« die entsprechende Stellung der letzten Bemerkung.
Die Zwischenvariable »zw« bestimmt den Antwortenbereich, d.h. die Differenz der Feldgrenzen »r(k)« und »n(k)«. Aus diesem Bereich wird dann in 3030 und 3040 der Antwortvariablen »aw« eine Antwort oder ein Kommentar zugewiesen.

Die Erweiterung des Restsatzes

Wenn Sie sich das Kommentar- und Antwortenvokabular im Scriptteil von ELIZA anschauen, werden Sie feststellen, daß viele Sätze einen Marker, dargestellt durch das Zeichen »*«, enthalten. In Kommentaren, die einen solchen Marker beinhalten, wird der Restsatz Ihrer Eingabe mit eingefügt. Dazu muß ELIZA zuerst feststellen, ob der vorgewählte Kommentarsatz im Feldelement »a$(aw)« einen Marker enthält:

```
3100 REM Restsatz anfuegen?
3120     FOR i=1 TO LEN(a$(aw))
```

```
3130        IF MID$(a$(aw),i,1)="*" THEN 3200
3140     NEXT i
3150 RETURN
3160 REM   kein Restsatz und
3170 REM   keine Konjugation
```

Wurde kein »*«-Zeichen gefunden, so springt das Programm in den Programmkopf zurück und wechselt dann direkt in den Ausgabeteil über. Andernfalls wird der Restsatz Ihrer Eingabe isoliert und in der Variablen »re$« abgelegt.

Gleichzeitig erhält der Restsatz-Marker »mk« den Wert 99 zugewiesen. Er wird benötigt, um der Kommentarausgabeprozedur mitzuteilen, daß der Marker »*« durch den Restsatz ersetzt werden soll.

Das sieht im Programm wie folgt aus:

```
3300 REM Restsatz isolieren
3310   re$=MID$(i$,p+LEN(zk$),LEN(i$))
3350   mk=99
3360   REM Restsatz in Antwort anfuegen
3510 :
3980 RETURN
```

Die Ausgabeprozedur

Es gibt zwei mögliche Ausgabeformen. Je nach Inhalt des Restsatz-Markers »mk« wird entweder eine normale Antwort oder eine Frage des Antwortfeldes »a$(aw)« angezeigt. Der jeweilige Kommentar ist ja bereits in den vorigen Prozeduren vorgewählt worden.

Soll an eine Antwort ein Restsatzteil angehängt werden, so wird die Antwort ohne den noch enthaltenen Marker »*« zusammen mit dem Satzrest Ihrer Eingabe ausgedruckt:

```
4000 REM Satzausgabe
4010 :
4100  PRINT " --> "
4120   IF mk=99 THEN GOTO 4200
4130   IF mk=0 THEN GOTO 4300
4140 :
4200  PRINT LEFT$(a$(aw),LEN(a$(aw))-1);re$
4250  RETURN
4260 :
4300  PRINT a$(aw)
4350  RETURN
```

Das komplette ELIZA-Programm

Damit ist ELIZA fertig. Unten sehen Sie das komplette Programm in der richtigen Reihenfolge der einzelnen bereits erörterten Module:

```
3 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel :  Eliza Psychiater
5 :
```

```
6 MODE 2
7 WIDTH 60
10 :
30 PAPER 3
40 PEN 0
50 CLS
60 :
100 GOSUB 30000
110      REM Initialisation
150 GOSUB 20000
160      REM Titel
170 :
200      GOSUB 1000
210       REM Dialog Satzeingabe
300      GOSUB 2000
310       REM Suchroutine
400      GOSUB 3000
410       REM Konjugation
500      GOSUB 4000
510       REM Antwortsatz
610 GOTO 200
989 STOP
990 :
1000 REM Dialog Satzeingabe
1100  PRINT
1110   INPUT "___> ";i$
1120    IF i$="" THEN 1110
1125    IF i$="stop" THEN STOP
1130    IF LEN(i$)<2 OR LEN(i$)>200 THEN PRINT
        "Ungueltige Eingabe...":GOTO 1110
1800 RETURN
1990 :
2000 REM Suchschleife
2010 :
2020 REM Schluesselwort waehlen
2090 ma=0
2100   FOR i=1 TO 38
2110      zk$=k$(i)
2120        GOSUB 2500
2125          IF ma=9 THEN 2150
2130   NEXT i
2140 :
2150 RETURN
2160 :
2500 REM Scanning
2510   FOR p=1 TO LEN(i$)
2520      w$=MID$(i$,p,LEN(zk$))
2530         IF w$=k$(i) THEN ma=9:k=i:RETURN
2540   NEXT p
2550   k=38:REM kein Keywort
2560 RETURN
2570 :
3000 REM Konjugation
3010 REM Antwortsatz vorwaehlen
3020     zw=n(k)-r(k)
3030     zr=INT(RND(1)*zw)
```

```
3040    aw=r(k)+zr
3045    IF aw=aalt THEN 3000
3047    aalt=aw
3050 :
3060 mk=0
3070 REM Restsatzmarker reset
3080 :
3100 REM Restsatz anfuegen?
3120     FOR i=1 TO LEN(a$(aw))
3130        IF MID$(a$(aw),i,1)="*" THEN 3200
3140     NEXT i
3150 RETURN
3160 REM   kein Restsatz und
3170 REM   keine Konjugation
3200 :
3300 REM Restsatz isolieren
3310    re$=MID$(i$,p+LEN(zk$),LEN(i$))
3350    mk=99
3360    REM Restsatz in Antwort anfuegen
3510 :
3980 RETURN
3995 :
4000 REM Satzausgabe
4010 :
4100  PRINT " --> "
4120    IF mk=99 THEN GOTO 4200
4130    IF mk=0 THEN GOTO 4300
4140 :
4200  PRINT LEFT$(a$(aw),LEN(a$(aw))-1);re$
4250  RETURN
4260 :
4300  PRINT a$(aw)
4350  RETURN
5000 :
20000 REM Titel
20010    PRINT
20020    PRINT TAB(12)" E L I Z A  ______  Psychiater"
20030    PRINT
20040    PRINT TAB(17)"(c) Olaf Hartwig  1985"
20050    PRINT
20060    WINDOW 6,79,6,25
20070    PAPER 2
20075    PEN 3
20080    CLS
20100 RETURN
29990 :
30000 REM Initialisation
30010      DIM r(38): REM Codezahl 1
30020      DIM n(38): REM Codezahl 2
30030      DIM k$(38):REM Keywoerter
30040      DIM a$(118):REM Antworten
30100 :
31000 REM Codezahlen lesen
31005     FOR i=1 TO 38
31010         READ r(i)
31020         READ n(i)
```

```
31050    NEXT i
31060 :
31100 REM data r(i),  n(i)
31110 DATA       1  , 3
31120 DATA       4  , 5
31130 DATA       6  , 9
31140 DATA       6  , 9
31150 DATA       10 , 13
31160 DATA       14 , 16
31170 DATA       17 , 19
31180 DATA       20 , 21
31190 DATA       22 , 24
31200 DATA       25 , 27
31210 DATA       28 , 31
31220 DATA       28 , 31
31230 DATA       32 , 34
31240 DATA       35 , 39
31250 DATA       40 , 48
31260 DATA       40 , 48
31270 DATA       40 , 48
31280 DATA       40 , 48
31290 DATA       40 , 48
31300 DATA       40 , 48
31310 DATA       49 , 50
31320 DATA       51 , 54
31330 DATA       55 , 58
31340 DATA       59 , 62
31350 DATA       63 , 63
31360 DATA       63 , 63
31370 DATA       64 , 68
31380 DATA       69 , 73
31390 DATA       74 , 75
31400 DATA       76 , 79
31410 DATA       80 , 82
31420 DATA       83 , 89
31430 DATA       90 , 92
31440 DATA       93 , 98
31450 DATA       99 , 105
31460 DATA       116, 118
31470 DATA       113, 115
31480 DATA       106, 112
31490 :
32000 REM Keywoerter lesen
32010    FOR i=1 TO 38
32020       READ k$(i)
32030    NEXT i
32070 :
32100 REM data Keywoerter
32110 DATA        kannst du
32120 DATA        kann ich
32130 DATA        bist du
32140 DATA        du bist
32150 DATA        ich kann
32160 DATA        ich fuehle
32170 DATA        kannst du
32180 DATA        kann ich
```

```
32190 DATA       bist du
32200 DATA       ich kann
32210 DATA       ich bin
32220 DATA       im
32230 DATA       du
32240 DATA       ich will
32250 DATA       was
32260 DATA       wie
32270 DATA       wer
32280 DATA       wo
32290 DATA       wann
32300 DATA       warum
32310 DATA       name
32320 DATA       grund
32330 DATA       leid
32340 DATA       traum
32350 DATA       hallo
32360 DATA       tag
32370 DATA       mag sein
32380 DATA       nein
32390 DATA       dein
32400 DATA       immer
32410 DATA       denke
32420 DATA       gleich
32430 DATA       ja
32440 DATA       freund
32450 DATA       computer
32460 DATA       eliza
32470 DATA       geld
32480 DATA       kein keywort
32490 :
33000 REM Antworten lesen
33010    FOR i=1 TO 118
33020        READ a$(i)
33030    NEXT i
33080 :
33100 REM Data antworten
33110 DATA glaubst du nicht daran dass ich das kann und zwar*
33120 DATA vielleicht wuerdest du das gerne tun naemlich*
33130 DATA du haettest es gerne wenn ich das koennte also
33140 DATA vielleicht moechtest du auch nicht*
33150 DATA moechtest du gerne*
33160 DATA warum glaubst du das ich das bin also*
33170 DATA gefaellt dir der gedanke dass ich so bin?
33180 DATA vielleicht wuerdest du gerne so sein also*
33190 DATA wuenscht du dir manchmal du waerst*
33200 DATA bist du nicht vielleicht*
33210 DATA warum bist du nicht*
33220 DATA moechtest du gerne dazu faehig sein das zu tun also*
33230 DATA beunruhigt dich das sehr stark oder weniger?
33240 DATA erzaehle mir doch etwas mehr ueber solche gefuehle.
33250 DATA fuehlst du dich oft*
33260 DATA geniest du dich so zu fuehlen?
33270 DATA glaubst du vielleicht ich bin nicht oder kann nicht*
33280 DATA vielleicht in besseren zeiten werde ich*
33290 DATA moechtest du dass ich*
```

```
33300 DATA glaubst du ich sollte faehig sein zu*
33310 DATA warum kannst du nicht*
33320 DATA warum bist du so daran interessiert ob oder ob
           ich nicht*
33330 DATA wuerdest du es vorziehen wenn ich nicht*
33340 DATA vielleicht bin ich in deiner phantasie*
33350 DATA woher weisst du dass ich das nicht kann?
33360 DATA hast du das schon versucht?
33370 DATA vielleicht kannst du nun*
33380 DATA kamst du zu mir weil du so bist also*
33390 DATA wie lange bist du schon*
33400 DATA glaubst du es ist normal so zu sein naemlich*
33410 DATA magst du das?
33420 DATA ich dachte wir spraechen ueber dich nicht ueber mich.
33430 DATA oh ich*
33440 DATA du sprichst da wirklich ueber mich nicht wahr?
33450 DATA was wuerde es fuer dich bedeuten wenn du das
           bekommen koenntest?
33460 DATA warum moechtest du*
33470 DATA angenommen du erhaelst bald*
33480 DATA was wuerde es fuer dich bedeuten wenn du das bekaemst?
33490 DATA ich wensche mir auch manchmal*
33500 DATA warum fragst du das?
33510 DATA interessiert dich diese frage sehr stark?
33520 DATA welche antwort gefiele dir am meisten?
33530 DATA was glaubst du?
33540 DATA kommen dir solche fragen oefter in den sinn?
33550 DATA was willst du eigentlich wissen?
33560 DATA hast du das schon andere personen gefragt?
33570 DATA hast du solche fragen schon mal zuvor gestellt?
33580 DATA was kommt dir sonst noch in den sinn wenn du
           das fragst?
33590 DATA namen interessieren mich weniger.
33600 DATA ich kuemmere mich nicht um namen...bitte fahre fort...
33610 DATA ist das der wirkliche grund oder verschweigst
           du mir etwas?
33620 DATA gaebe es vielleicht andere gruende-denk mal kurz nach...
33630 DATA erklaert dieser grund unter umstaenden noch
           etwas anderes?
33640 DATA welche andere gruende koennte es geben?
33650 DATA bitte entschuldige dich nicht!
33660 DATA entschuldigen sind nicht noetig...wirklich nicht!
33670 DATA wie fuehlst du dich wenn du dich entschuldigst?
33680 DATA sei doch nicht so defensiv!
33690 DATA was verraet dir dieser traum...was bedeutet er dir?
33700 DATA traeumst du eigentlich oft oder kommt das selten vor?
33710 DATA welche personen und welche gegenstaende erscheinen
           in deinen traeumen
33720 DATA beunruhigen dich solche traeume?
33730 DATA hallo wie geht es dir...erzaehle mir dein problem.
33740 DATA du bist dir da scheinbar nicht ganz sicher...gibt
           es dafuer gruende?
33750 DATA warum dieser unsichere ton?
33760 DATA kannst du nicht positiver denken?
33770 DATA du bist dir wirklich nicht sicher?
33780 DATA weist du nicht warum?
```

```
33790 DATA warum keine*
33800 DATA sag doch nicht immer nein...es ist so negativ.
33810 DATA warum nicht?
33812 DATA bist du dir sicher?
33814 DATA wirklich nicht?
33820 DATA warum bist du besorgt ueber mein*
33830 DATA wie steht es eigentlich mit deinem*
33840 DATA kannst du dabei an ein spezielles beispiel denken?
33850 DATA wann?
33860 DATA woran denkst du?
33870 DATA wirklich immer?
33880 DATA denkst du wirklich so?
33890 DATA aber du bist dir nicht sicher dass du*
33900 DATA du bezweifelst dass du*
33910 DATA in welcher form?
33920 DATA welche auswirkungen siehst du?
33930 DATA was zeigt dir dieser zusammenhang?
33940 DATA welche anderen verbindungen siehst du?
33950 DATA koennten da wirklich gemeinsamkeiten bestehen?
33960 DATA wie?
33970 DATA du scheinst dir ja ganz sicher zu sein!
33980 DATA bist du dir da auch sicher?
33990 DATA ich kann das nachvollziehen.
34000 DATA ich verstehe.
34010 DATA warum erwaehnst du freunde?
34020 DATA beunruhigen dich deine freunde?
34030 DATA aergern dich deine freunde?
34040 DATA bist du dir sicher dass du ueberhaupt freunde besitzt?
34050 DATA vielleicht liegt der grund deiner probleme bei
           deinen freunden?
34060 DATA vielleicht haben deine freunde negative
           auswirkungen auf dich.
34070 DATA beunruhigen dich computer?
34080 DATA sprichst du dabei eigentlich von mir im besonderen?
34090 DATA hast du angst vor maschinen?
34100 DATA hast du da gerade computer erwaehnt?
34110 DATA haben maschinen mit deinem problem zu tun?
34120 DATA glaubst du nicht dass computer den menschen
           helfen koennen
34130 DATA was an den maschinen stoert dich im besonderen?
34140 DATA sag hast du irgendwelche psychologischen probleme?
34150 DATA was sagt dir das? hat das eine bedeutung fuer dich?
34160 DATA aha! fahre bitte fort...
34170 DATA ich bin mir da nicht so ganz sicher ob ich dir
           da folgen kann...
34180 DATA kannst du deine gedanken etwas weiter ausfuehern?
34190 DATA wuerdest du das vielleicht etwas naeher beschreiben?
34200 DATA das ist ja interessant!
34210 DATA warum hast du probleme mit deinen mitmenschen?
34220 DATA glaubst du eigentlich dass geld alles bedeutet?
34230 DATA bist du dir sicher dass geld dein problem ist?
34240 DATA ich dachte wir sprachen ueber dich...nicht ueber mich!
34250 DATA was soll mit mir sein?
34260 DATA warum erwaehnst du immer meinen namen?
34270 :
36000 RETURN
```

Ein Demonstrationslauf von ELIZA als Psychiater

Was leistet das ELIZA-Programm? Schauen wir uns dazu einmal einen Teil eines Beispieldialogs an:

ANWENDER:	Guten Tag!
ELIZA:	Hallo wie geht es dir... ...erzähle mir dein Problem.
ANWENDER:	Ich bin depressiv.
ELIZA:	Wie lange bist du schon depressiv?
ANWENDER:	Seit einem Monat.
ELIZA:	Was sagt dir das? Hat das eine Bedeutung für dich?
ANWENDER:	Um das zu erfahren, kam ich ja zu dir.
ELIZA:	Bist du dir da auch ganz sicher?
ANWENDER:	Ja absolut sicher.
ELIZA:	Ich kann das nachvollziehen.
ANWENDER:	Wie kann ich meine Depressionen überwinden?
ELIZA:	Vielleicht möchtest du auch nicht meine Depressionen überwinden?
ANWENDER:	Wie kommst du darauf?
ELIZA:	Hast du solche Fragen schon mal zuvor gestellt?
	.
	.
	.

Der Dialog läuft kontinuierlich so weiter. Dabei macht ELIZA im Rahmen der bescheidenen Möglichkeiten des CPC und der Komplexität der deutschen Grammatik als Dialogpartner einen verhältnismäßig kompetenten Eindruck.
Bedingt durch die trotz des großen Umfangs des Listings doch recht einfache Struktur des Programms hat es gewisse Grenzen. Sie sind vor allem dadurch bedingt, daß die deutsche Grammatik, die bei der Festlegung der Ausgabe berücksichtigt werden muß, relativ kompliziert ist. Trotzdem lassen sich doch recht befriedigende Ergebnisse erzielen.

2.4.2 Das ELIZA-VORG-Programm

Ich möchte Sie ermuntern, selber mit ELIZA zu experimentieren und das Programm zu erweitern. Als Anregung stelle ich Ihnen hier ein weiteres ELIZA-Programm vor. Es besitzt ein neues Scriptmodul. Beachten Sie auch die kleinen, aber wichtigen Änderungen des Steuerungsmoduls.
ELIZA stellt in der neuen Version einen barschen, gleichzeitig aber auch ein wenig schizophrenen Vorgesetzten dar.
Hier kommt das komplette Programm:

```
3 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel : Eliza Vorgesetzter
5 :
6 MODE 2
7 WIDTH 60
10 :
30 PAPER 3
40 PEN 0
50 CLS
70 :
100 GOSUB 30000
110      REM Initialisation
150 GOSUB 20000
160      REM Titel
170 :
200      GOSUB 1000
210       REM Dialog Satzeingabe
300      GOSUB 2000
310       REM Suchroutine
400      GOSUB 3000
410       REM Konjugation
500      GOSUB 4000
510       REM Antwortsatz
610 GOTO 200
989 STOP
990 :
1000 REM Dialog Satzeingabe
1100  PRINT
1110   INPUT "_ _ _> ";i$
1120    IF i$="" THEN 1110
1125    IF i$="stop" THEN STOP
1130    IF LEN(i$)<2 OR LEN(i$)>200 THEN PRINT
        "Ungueltige Eingabe...":GOTO 1110
1800 RETURN
1990 :
2000 REM Suchschleife
2010 :
2020 REM Schluesselwort waehlen
2090 ma=0
2100   FOR i=1 TO 46
2110      zk$=k$(i)
2120        GOSUB 2500
2125          IF ma=9 THEN 2150
2130   NEXT i
2140 :
2150 RETURN
2160 :
2500 REM scanning
2510   FOR p=1 TO LEN(i$)
2520      w$=MID$(i$,p,LEN(zk$))
2530        IF w$=k$(i) THEN ma=9:k=i:RETURN
2540   NEXT p
2550   k=45:REM kein Keywort
2560 RETURN
2590 :
3000 REM Konjugation
```

```
3010 REM Antwortsatz vorwaehlen
3020     zw=n(k)-r(k)
3030     zr=INT(RND(1)*zw)
3040   aw=r(k)+zr
3045   IF aw=aalt THEN 3000
3047   aalt=aw
3050 :
3060 mk=0
3070 REM Restsatzmarker reset
3080 :
3100 REM Restsatz anfuegen?
3120    FOR i=1 TO LEN(a$(aw))
3130       IF MID$(a$(aw),i,1)="*" THEN 3200
3140    NEXT i
3150 RETURN
3160 REM   kein Restsatz und
3170 REM   keine Konjugation
3200 :
3300 REM Restsatz isolieren
3310   re$=MID$(i$,p+LEN(zk$),LEN(i$))
3350   mk=99
3360   REM Restsatz in Antwort anfuegen
3510 :
3980 RETURN
3990 :
4000 REM Satzausgabe
4010 :
4100  PRINT " --> "
4120   IF mk=99 THEN GOTO 4200
4130   IF mk=0 THEN GOTO 4300
4140 :
4200  PRINT LEFT$(a$(aw),LEN(a$(aw))-1);re$
4250  RETURN
4260 :
4300  PRINT a$(aw)
4350  RETURN
5000 :
20000 REM Titel
20010   CLS
20015   PRINT
20020   PRINT TAB(12)" E L I Z A    V O R G."
20030   PRINT
20040   PRINT TAB(17)"(c) Olaf Hartwig  1985"
20060   WINDOW 6,79,6,25
20070   PAPER 2
20080   PEN 3
20090   CLS
20100 RETURN
28000 :
30000 REM Initialisation
30010     DIM r(46): REM Codezahl 1
30020     DIM n(46): REM Codezahl 2
30030     DIM k$(46):REM Keywoerter
30040     DIM a$(121):REM Antworten
30100 :
31000 REM Codezahlen lesen
```

```
31005   FOR i=1 TO 45
31010       READ r(i)
31020       READ n(i)
31030   NEXT i
31060 :
31100 REM data r(i), n(i)
31110 DATA       1  , 3
31120 DATA       4  , 6
31130 DATA       7  , 10
31140 DATA       11 , 15
31150 DATA       11 , 15
31160 DATA       16 , 20
31170 DATA       21 , 24
31180 DATA       25 , 27
31190 DATA       28 , 35
31200 DATA       28 , 35
31210 DATA       28 , 35
31220 DATA       28 , 35
31230 DATA       28 , 35
31240 DATA       28 , 35
31250 DATA       36 , 40
31260 DATA       36 , 40
31270 DATA       36 , 40
31280 DATA       41 , 45
31290 DATA       41 , 45
31300 DATA       41 , 45
31310 DATA       41 , 45
31320 DATA       41 , 45
31330 DATA       46 , 48
31340 DATA       49 , 53
31350 DATA       54 , 57
31360 DATA       58 , 63
31370 DATA       64 , 69
31380 DATA       64 , 69
31390 DATA       70 , 74
31400 DATA       70 , 74
31410 DATA       75 , 77
31420 DATA       75 , 77
31430 DATA       78 , 85
31440 DATA       86 , 89
31450 DATA       90 , 94
31460 DATA       90 , 94
31470 DATA       95 , 98
31480 DATA       95 , 98
31490 DATA       99 , 101
31500 DATA       102, 106
31510 DATA       107, 108
31520 DATA       109, 111
31530 DATA       112, 113
31540 DATA       112, 113
31550 DATA       114, 121
31590 :
32000 REM Keywoerter lesen
32010    FOR i=1 TO 46
32020        READ k$(i)
32030    NEXT i
```

```
32040 :
32100 REM data Keywoerter
32110 DATA       ich bin
32120 DATA       ich kann
32130 DATA       will
32140 DATA       wuensche
32150 DATA       wunsch
32160 DATA       du
32170 DATA       sie
32180 DATA       immer
32190 DATA       warum
32200 DATA       wie
32210 DATA       wer
32220 DATA       was
32230 DATA       wo
32240 DATA       wann
32250 DATA       nein
32260 DATA       nicht
32270 DATA       nie
32280 DATA       ja
32290 DATA       stimmt
32300 DATA       immer
32310 DATA       genau
32320 DATA       richtig
32330 DATA       grund
32340 DATA       traum
32350 DATA       hallo
32360 DATA       denke
32370 DATA       geld
32380 DATA       gehalt
32390 DATA       gleich
32400 DATA       sofort
32410 DATA       ende
32420 DATA       freizeit
32430 DATA       freund
32440 DATA       computer
32450 DATA       arbeit
32460 DATA       beruf
32470 DATA       idiot
32480 DATA       trottel
32500 DATA       streik
32520 DATA       moegen
32530 DATA       spielen
32540 DATA       alt
32550 DATA       chef
32560 DATA       boss
32570 DATA       kein keywort
32590 :
33000 REM Antworten lesen
33010     FOR i=1 TO 121
33020         READ a$(i)
33030     NEXT i
33070 :
33100 REM Data Antworten
33110 DATA sind sie sich da auch ganz sicher dass sie das
           wirklich sind und zwar*
```

```
33120 DATA vielleicht wuerden sie das gerne sein naemlich*
33130 DATA ich glaube ihnen kein wort das sie das sind-*
33140 DATA schoen wenn sie das koennen!
33150 DATA reden sie nicht so viel-fangen sie an!
33160 DATA dann setzen sie ihr koennen sinnvoll ein und
           belaestigen mich nicht!
33170 DATA was ist das fuer ein tonfall.sie wollen nicht
           sondern moechten.
33180 DATA was wollen sie ?!!
33190 DATA ihnen geht es wohl zu gut!
33200 DATA abgelehnt! sprechen sie nicht weiter!
33210 DATA was wuenschen sie? *
33220 DATA habe ich mich da verhoert? sie wuenschen*
33230 DATA jeder wunsch muss hier hart erarbeitet werden!
33240 DATA sie wuenschen sich da vielleicht etwas zu viel!
33250 DATA ich habe ihre wuensche allmaehlich gestrichen satt!
33260 DATA gewoehnen sie sich einen anderen tonfall an!
33270 DATA siezen sie mich gefaelligst!
33280 DATA wenn sie mich nicht augenblicklich mit 'sie'
           anreden fliegen sie raus
33290 DATA hoeren sie sofort mit ihrer duzerei auf!
33300 DATA wagen sie es nicht noch einmal 'du' zu mir zu sagen!
33310 DATA reden sie schon wieder von mir?
33320 DATA beziehen sie das auf mich?
33330 DATA ihr siezen gefaellt mir.
33340 DATA warum wenden sie sich immer an mich?
33350 DATA wirklich immer?
33360 DATA immer noch?
33370 DATA warum?
33380 DATA wesshalb fragen sie ?
33390 DATA sie haben meine geduld bereits ueberstrapaziert!
           keine fragen mehr!
33400 DATA noch eine weitere frage und sie fliegen raus!
33410 DATA ihre ewige fragerei ist ja nicht zum aushalten!
33420 DATA stellen sie keine so idiotischen fragen!
33430 DATA schluss mit der fragerei!
33440 DATA hier wird nicht gefragt sondern gespurrt.
33450 DATA hier stelle ich die fragen.
33460 DATA sagen sie nicht immer nein!
33470 DATA noch ein 'nein' und...
33480 DATA hier wird positiv gedacht!
33490 DATA wirklich nicht?
33500 DATA was soll das heissen?
33510 DATA prima!
33520 DATA jawohl! denken sie positiv!
33530 DATA heisst das sie sind mit mir einer meinung?
33540 DATA sie denken also positiv darueber?
33550 DATA es gefaellt mir wenn sie positiv denken.
33560 DATA suchen sie nicht nach gruenden sondern arbeiten sie!
33570 DATA ihre gruende interessieren mich nicht!
33580 DATA behalten sie ihre gruende besser fuer sich!
33590 DATA haben sie etwa bei ihrer arbeit getraeumt?
33600 DATA sie haben doch wohl von mir getraeumt?
33610 DATA ihre traeume interessieren mich nicht!
33620 DATA kommen sie mir nicht mit solchen dingen!
33630 DATA traeumen koennen sie zu hause aber nicht bei der arbeit!
```

```
33640 DATA guten tag heisst das!
33650 DATA was erlauben sie sich!
33660 DATA sagen sie nicht noch einmal hallo zu mir!
33670 DATA wie sprechen sie mit ihrem vorgesetzten?
33680 DATA sie denken zu viel.
33690 DATA verschwenden sie ihre energie nicht mit zu vielem
           denken!
33700 DATA denken sie nicht - arbeiten sie!
33710 DATA wenn sie das noch einmal denken fliegen sie raus!
33720 DATA genug gedacht fuer heute!
33730 DATA sie werden nicht fuers denken bezahlt sondern
           fuers arbeiten!
33740 DATA sie wollen etwa schon wieder mehr geld?
33750 DATA erwaehnen sie noch einmal das wort geld und ich
           kuerze ihr gehalt!
33760 DATA warum kommen sie auf geld?
33770 DATA geld geht mich nichts an.
33780 DATA geld erhaelt nur der der arbeitet ... sie nicht!
33790 DATA aergern sie mich nicht mit ihren ewigen gedanken
           an geld!
33800 DATA sagen sie das nicht sondern handeln sie!
33810 DATA lassen sie dem satz taten folgen!
33820 DATA handeln sie endlich!
33830 DATA ihre ausfluechte reichen mir...tun sie etwas.
33840 DATA nicht gleich oder sofort sondern jetzt! keine
           ausfluechte!
33850 DATA wie koennen sie jetzt an ihre freizeit denken!
33860 DATA sie muessen heute ueberstunden machen...keine freizeit!
33870 DATA sie haben doch erst gerade ferien gehabt.
33880 DATA ihr einziger freund bin ich!
33890 DATA warum denken sie an freunde?
33900 DATA uebrigens ihre freunde gehen mir auf die nerven.
33910 DATA wie koennen sie jetzt an freunde denken...sie
           muessen arbeiten.
33920 DATA belaestigen sie mich nicht immer mit ihren freunden.
33930 DATA sie haben jetzt keine zeit fuer freunde!
33940 DATA denken sie an etwas sinnvolleres!
33950 DATA haben sie ueberhaupt freunde?
33960 DATA was halten sie von computern?
33970 DATA haben sie etwa etwas gegen computer?
33980 DATA computer sind sinnvoll. Oder was finden sie?
33990 DATA wie bitte? computer sind *
34000 DATA wie ist denn ihre arbeitseinstelung?
34010 DATA was? arbeit ist*
34020 DATA wie gefaellt ihnen denn ihre arbeit?
34030 DATA haben sie zu wenig arbeit?
34040 DATA sie moechten wohl mehr in ihrem beruf arbeiten?
34050 DATA reden sie etwa von mir?
34060 DATA was haben sie da eben gesagt?
34070 DATA meinen sie etwa mich?
34080 DATA ein weiteres wort und ich feure sie...fristlos!
34090 DATA streik??
34100 DATA schon wieder ein streik?
34110 DATA bei streiks hilft nur aussperrung oder entlassungen!
34120 DATA was moegen sie?...*
34130 DATA warum moegen sie*
```

```
34140 DATA moegen sie auch ihre arbeit?
34150 DATA sie moegen doch sicher ganz besonders mich nicht wahr?
34160 DATA sie moegen mir entschieden zu viel!
34170 DATA warum spielen sie das gerne und zwar*
34180 DATA bei mir wird nicht gespielt!
34190 DATA wer ist hier alt?
34200 DATA sie werden auch allmaehlich alt.
34210 DATA finden sie mich etwa auch alt?
34220 DATA wie reden sie mit mir?
34230 DATA was bilden sie sich ein mich so zu nennen!
34240 DATA was wollen sie eigentlich?
34250 DATA halten sie gefaelligst ihren mund!
34260 DATA nuscheln sie nicht so!
34270 DATA gewoehnen sie sich einen anderen tonfall an!
34280 DATA ist das alles was sie mir zu sagen haben?
34290 DATA ich muss noch etwas erledigen. warten sie
           einige minuten!
34300 DATA entschuldigen sie mich einige minuten. ich muss
           kurz telephonieren.
34310 DATA starren sie mich nicht so aufdringlich an!
34320 DATA
34330 DATA
34340 :
36000 RETURN
```

Ein Beispieldialog

Werfen wir abschließend einen Blick auf einen Beispieldialog des neuen ELIZA-Programms:

ANWENDER: Haben Sie mich gerufen?
ELIZA: Halten Sie gefälligst Ihren Mund!
ANWENDER: Warum denn?
ELIZA: Hier wird nicht gefragt sondern gespurt!
ANWENDER: Ich mag dieses Motto nicht!
ELIZA: Wirklich nicht?
ANWENDER: Ganz sicher nicht!
ELIZA: Noch ein »nein« und ...
ANWENDER: Was »und«?
ELIZA: Schluß mit der Fragerei!

.
.
.

Auch das ELIZA-Programm mit dem neuen Scriptteil ist fähig, sich verhältnismäßig gut mit dem Anwender zu unterhalten. Da aber der Scripttext teilweise sicher nicht realitätsbezogen abgefaßt wurde, wirkt der Dialog oft überzogen.

Der Stil und auch das Sachgebiet des Dialogs lassen sich jedoch von Ihnen beliebig und auch erheblich realistischer gestalten. Das Programm sollte nur zur Veranschaulichung dienen. Es ist auch möglich, ein ELIZA-Programm mit Programmen vom Typ RACTER und MICRO-DBABY zu kombinieren.
Ich wünsche Ihnen viel Spaß bei eigenen Experimenten!

3 Computer-Kreativität

Nachdem bisher die Erkennung und Dekodierung natürlicher Sprache und die Erzeugung sinnbezogener Antworten im Mittelpunkt der Betrachtungen standen, wenden wir uns nun der Generierung von Texten durch den Computer zu.
Anhand des Programms SCRUDU, das automatisch Gedichte generiert, werden Grundprinzipien menschlicher und maschineller Kreativität aufgezeigt. Ausführlich behandelt wird dabei das grammatische Format von Sätzen.
Die bisher erarbeiteten Techniken sind grundlegend für die Erzeugung von grammatisch korrekten Antworten auf einer höheren Ebene, als sie durch die bisher vorgestellten Techniken möglich ist.

3.1 Was ist Kreativität?

Können Computer kreativ sein? Diese Frage ist wohl noch umstrittener als die der Computer-Intelligenz. Beschäftigen wir uns vorerst kurz mit der Frage der menschlichen Kreativität. Wie vollbringt der Mensch kreative Leistungen?
Über diesen Komplex sind von Psychologen unzählige Bücher geschrieben worden. Es läßt sich trotz unterschiedlichster Ansichten von Experten auf diesem Gebiet eine klare Linie ziehen:

Voraussetzung für Kreativität ist eine Wissensbank. Das im menschlichen Gehirn gespeicherte Wissen wird dann mit neu aufgenommenen Informationen verglichen und erzeugt Assoziationen.

Ein gutes Beispiel für diesen Vorgang ist die Verarbeitung über das Auge aufgenommener Bilder: »das sieht doch aus wie...«. Den Ausspruch haben Sie sicher schon gehört. Das Gehirn erhält dabei Informationen von einem über die Augen wahrgenommenen Bild und vergleicht es mit anderen bereits bekannten Bildern. Hierbei werden Assoziationen erzeugt, die dann zu einem neuen Begriff zusammengesetzt werden.

Auf einen Nenner gebracht kann man sagen: Kreativität ist die Synthese mehrerer bisher unverbundener Informationen in einer bestimmten Weise.

Bei einer Melodie werden Noten in einer bestimmten Weise miteinander verknüpft. Ein anderes Beispiel ist der sich ständig verändernde Umgangswortschatz. Neue Wörter entstehen meistens durch Zusammensetzen mehrerer, bisher unverbundener Vokabeln zu einem neuen Zusammenhang.
Das hier vorgestellte Programm SCRUDU funktioniert nach diesem Grundprinzip. Die nach unterschiedlichen Mustern hergestellten Verse werden aus einzelnen Wortgruppen zusammengesetzt, die einen neuen Sinn ergeben. Dabei müssen die jeweiligen Komponenten eines Verses einen sinnvollen Zusammenhang bilden. Das Grundvokabular des Programms muß also entsprechend vorgewählt werden.
Ein anderes Problem ist die Bewältigung des korrekten grammatischen Formats. Dieses Problem ist beim komplizierten grammatischen Aufbau der deutschen Sprache besonders schwierig zu lösen, aber es ist – wenn auch mit gewissen Einschränkungen – durchführbar. Wie das genau zu realisieren ist, wird im Programm noch näher erläutert.

3.2 Das Generatorprogramm SCRUDU

Das Programm besteht aus insgesamt acht Unterprozeduren. Davon werden die ersten beiden nur einmal zu Beginn des Programms aufgerufen, die übrigen sechs bilden eine Schleife, die jeweils unterschiedliche Verstypen generiert.
Der Kopfteil des Programms lautet folgendermaßen:

```
3 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel : SCRUDU - Gedichtgenerierung
5 :
6      MODE 2
7      BORDER 9,9
8      WIDTH 60
10 :
30     PAPER 2
40     PEN 1
50     CLS
60 :
100 REM ************ Supervisor ************
102 :
105        GOSUB 30000
110           REM Initialisation
160        GOSUB 40000
170           REM Titel
180 :
200       GOSUB 1000
210           REM Verstyp ermitteln
```

```
220 :
300       IF ty=1 THEN GOSUB 2000
310               REM Typ 1
400       IF ty=2 THEN GOSUB 3000
410               REM Typ 2
500       IF ty=3 THEN GOSUB 4000
510               REM Typ 3
600       IF ty=4 THEN GOSUB 5000
610               REM Typ 4
620 :
700        GOSUB 50000
710            REM weiterer Vers
720 GOTO 200
```

Die Initialisation

Dieser Programmteil dimensioniert fünf Felder, in denen das Grundvokabular abgelegt wird. Eingelesen werden Artikel, Adjektive, Substantive, Verben und Präpositionen. Der Teil sieht so aus:

```
30000 REM ********* Initialisation ***********
30010 :
30100 REM -------- Felder definieren ---------
30110  DIM a$(4)
30120     REM Artikel
30130  DIM ad$(50)
30140     REM Adjektive
30150  DIM n$(50)
30160     REM Nomen
30170  DIM v$(14)
30180     REM Verben
30190  DIM p$(6)
30200     REM Praeposition
30210 :
31000 REM ---------- Felder einlesen ---------
31010    REM Artikel
31020       FOR i=1 TO 4
31030          READ a$(i)
31040       NEXT i
31050 :
31100    REM Adjektive
31120       FOR i=1 TO 50
31130          READ ad$(i)
31140       NEXT i
31150 :
31200    REM Nomen
31220       FOR i=1 TO 50
31230          READ n$(i)
31240       NEXT i
31250 :
31300    REM Verben
31320       FOR i=1 TO 14
31330          READ v$(i)
31340       NEXT i
```

```
31360 :
31400    REM Praeposition
31420       FOR i=1 TO 6
31430          READ p$(i)
31440       NEXT i
31450 :
31500 RETURN
31510 :
32000 REM ------------- Daten ----------------
32010   REM Artikel -------------------------
32020     DATA der,das
32030     DATA ein,der
32080 :
33000   REM Adjektive -----------------------
33010     DATA junge,schwarze
33020     DATA wilde,verdorrte
33030     DATA umherschweifende,zinnoberrote
33040     DATA pochende,funkelnde
33050     DATA kleine,ruhige
33060     DATA rauhe,rote
33070     DATA alte,truebe
33080     DATA fruehe,schmale
33090     DATA frostige,gruene
33100     DATA spaete,lange
33110     DATA klare,sehnsuechtige
33120     DATA fallende,blaue
33130     DATA wiegende,feuchte
33140     DATA kalte,zerbrochene
33150     DATA klingende,gefleckte
33160     DATA feine,stille
33170     DATA verschneite,glaenzende
33180     DATA trockene,dunkle
33190     DATA schummrige,durchsichtige
33200     DATA neblige,leere
33210     DATA eisige,heisse
33220     DATA wolkige,warme
33230     DATA rauschende,toenende
33240     DATA herbstliche,reizende
33250     DATA beraubende,gelbe
33990 :
33995    REM Nomen -------------------------
34000     DATA Blume,Leuchtkaefer
34010     DATA Ruhe,Baum
34020     DATA Glitzern,Daemmerung
34030     DATA Funkeln,Sonne
34040     DATA See, Schneeflocke
34050     DATA Lichtung,Vogel
34060     DATA Feder,Dunst
34070     DATA Bach,Wald
34080     DATA Schatten,Nacht
34090     DATA Himmel,Brandung
34100     DATA Grass,Berg
34110     DATA Blume,Wasser
34120     DATA Geraeusch,Form
34130     DATA Feld,Tanne
34140     DATA Violett,Welle
```

```
34150     DATA Tau,Dunst
34160     DATA Blatt,Schatten
34170     DATA Schmetterling,Busch
34180     DATA Huegel,Wolke
34190     DATA Sonnenroete,Regen
34200     DATA Wind,Morgen
34210     DATA Meer,Schnee
34220     DATA Teich,Wasserfall
34230     DATA Brise,Fluss
34240     DATA Sonnenaufgang,Mond
34250 :
35000 REM Verben -----------------------------
35010     DATA ist gefallen,schleicht
35020     DATA faellt,ist getropft
35030     DATA trudelt,fliesst
35040     DATA raunt,schlaeft
35050     DATA schuettelt,treibt
35060     DATA hat gestoppt,kaempft
35070     DATA flattert,ist gestiegen
35080 :
36000 REM Praepositionen ---------------------
36010     DATA unter,ueber
36020     DATA nahe,an
36030     DATA in,auf
```

Die Bestimmung des Verstyps

In diesem Programm-Modul wird der Verstyp-Variablen »ty« ein zufälliger Wert für einen Verstyp zwischen eins und vier zugewiesen. Ist der Typ gleich dem des letzten Verses, wird die Zuweisung wiederholt.
Gleichzeitig wird der gewählte Verstyp in inverser Darstellung im Text-Window angezeigt:

```
1000 REM ********* Verstyp ermitteln *******
1050     ty=1+INT(RND(1)*4)
1070 :
1080       IF ta=ty THEN GOTO 1000
1100       ta=ty
1200 :
1205     PRINT
1207     PRINT TAB (50);
1210     PAPER 0: PEN 1: PRINT "(Verstyp___";ty;"__)"
1215     PAPER 1: PEN 0
1220 :
1400 RETURN
```

Der Versgenerierungs-Teil

Wir schauen uns beispielhaft für alle vier verschiedenen Verstypen die Erzeugung eines Verses des Typs 1 (ty=1) an.
Zunächst werden in der Unterroutine 1800 vor jedem neuen Vers das Satzfeld gelöscht und einige Marker neu gesetzt. Anschließend wählt SCRUDU aus seinen Vokabularfeldern ein Substantiv,

eine Präposition, einen Artikel und ein weiteres Nomen. Das geschieht in den Subroutinen 23000, 25000, 21000 sowie 23000. Die jeweils ermittelten Wörter werden im Feld »w$(n)« abgelegt. Dies wird in der Routine in Zeile 19000 mit der Satzvariablen »sa$« vorgenommen.
Nach dem Grundverfahren bestimmt SCRUDU drei unterschiedliche Sätze:

```
2000 REM ********* Verstyp Nr. 1 ***********
2003   GOSUB 23000
2005 :
2007 REM
2010 REM --------- erster Satz -------------
2015 :
2025 GOSUB 18000
2040      REM Nomen
2050   GOSUB 25000
2060      REM plus Praeposition
2070   GOSUB 21000
2080      REM plus Artikel
2090   GOSUB 23000
2100      REM plus Nomen
2140 :
2170 GOSUB 19000
2180   PRINT sa$
2195 :
2200 REM --------- zweiter Satz ------------
2210 :
2215 GOSUB 18000
2220    GOSUB 21000
2225       REM plus Artikel
2230    GOSUB 22000
2240       REM Adjektiv
2250    GOSUB 23000
2260       REM plus Nomen
2270 :
2275 GOSUB 19000
2390     PRINT sa$
2395 :
2400 REM --------- dritter Satz ------------
2420 :
2425 GOSUB 18000
2430   GOSUB 21000
2440     REM Artikel
2450   GOSUB 22000
2460     REM plus Adjektiv
2470   GOSUB 23000
2490     REM plus Nomen
2540 :
2570 GOSUB 19000
2590     PRINT sa$
2595 :
2600 RETURN
```

Die anderen vier Verstypen werden dementsprechend nach demselben Grundmuster erzeugt. Die einzelnen Programm-Module sind in dem einige Seiten weiter folgenden kompletten Listing des Programms SCRUDU aufgeführt.

Die Verstyp-Subroutinen

Die Hilfsroutinen sind bereits kurz angesprochen worden. Hier werden sie nun in ihrer vollständigen Form vorgestellt:

1. Satzfeld-Reset

Zunächst wird die Zählvariable des Feldes »w(n)« auf Null gesetzt. Die Marker für die Stellung des Artikels und des Adjektivs sowie der Präpositionsanzeiger werden gelöscht. Anschließend wird jedes Element des alten Satzfeldes »w$(n)« geleert.
Das sieht in der Programm-Realisierung folgendermaßen aus:

```
18000 REM ********* Satzfeld reset ***********
18050 n=0
18070 pa=0
18080 ar=0
18090 ad=0
18100    FOR i=0 TO 10
18110          w$(i)=""
18120    NEXT i
18130 RETURN
```

2. Die Satzzuweisung

Die Satzvariable »sa$« wird hier mit »n«-Elementen des Wortfeldes »w$()« belegt. Dabei werden die jeweiligen Worte durch Leerzeichen voneinander getrennt.
Das Modul lautet:

```
19000 REM ********* Satzzuweisung  ***********
19080 sa$="  ": REM zwei Leerzeichen!
19100  FOR i= 1 TO n
19110     sa$=sa$+w$(i)+" "
19120  NEXT i
19200 RETURN
```

3. Bestimmung von Wörtern aus dem Vokabular

Die sechs unabhängigen Unterprozeduren werden einzeln vom Versgenerierungsteil von SCRUDU aufgerufen. In jeder Routine wird zunächst die Anzahl der Wörter (n) eines Satzes um eins erhöht.
Bei Artikeln, Adjektiven und Präpositionen müssen zur Ermittlung der späteren grammatisch korrekten Form Hilfsmarker gesetzt werden. Anschließend wird je ein Element der jeweiligen Vokabulargruppe ausgewählt und dem Array »w$(n)« zugewiesen:

```
20000 REM * Woerter aus Vokabular bestimmen **
20990 :
21000 REM ------------- Artikel --------------
21050  n=n+1
21070  ar=n
```

```
21100    vo=1+INT((RND)*4)
21200    w$(n)=a$(vo)
21900 RETURN
21990 :
22000 REM ------------ Adjektive -------------
22050  n=n+1
22060  ad=n
22100    vo=1+INT(RND(1)*50)
22200    w$(n)=ad$(vo)
22900 RETURN
22910 :
23000 REM -------------- Nomen ---------------
23050  n=n+1
23100    vo=1+INT(RND(1)*50)
23200    w$(n)=n$(vo)
23900 RETURN
23940 :
24000 REM -------------- Verben --------------
24050  n=n+1
24100    vo=1+INT(RND(1)*14)
24200    w$(n)=v$(vo)
24900 RETURN
24990 :
25000 REM ---------- Praepositionen ----------
25050  n=n+1
25070  pa=99
25100    vo=1+INT(RND(1)*6)
25200    w$(n)=p$(vo)
25900 RETURN
```

Der Gedichtstitel

Zu jedem gesamten Gedicht wird ein Titel nach dem oben bereits erörterten Verfahren ermittelt. Der Titel wird in einem eigenen Window angezeigt.
Anschließend wird das Textfenster mit seinen Farben für die Versausgabe gesetzt:

```
41000 REM ----- Gedichtstitel waehlen ------
41100 GOSUB 18000
41110   GOSUB 21000
41120      REM Artikel
41130   GOSUB 22000
41140      REM plus Adjektiv
41150   GOSUB 23000
41160      REM plus Nomen
41180 :
41200 GOSUB 19000
41250 WINDOW 3,60,7,7:CLS
41300    PRINT "  Titel___>  ";sa$
41500 :
41510       PAPER 1
41520       PEN 0
41530 WINDOW 5,79,9,25
41540       CLS
42000 RETURN
```

Der nächste Vers

Betätigen Sie nach der Generierung eines dreizeiligen Verses in diesem Modul die »s«-Taste, so wird die Versgenerierung beendet. Jede andere Taste führt zur Erstellung eines weiteren Verses:

```
50000 REM *** Taste=weiterer Vers,"s"=Stop ***
50005 :
50010   KE$=INKEY$:IF KE$="" THEN 50010
50020    IF KE$="s" THEN 980
50030 RETURN
```

3.2.1 Das komplette SCRUDU-Programm

Im folgenden sehen Sie das komplette Programm im anschaulichen Zusammenhang mit dem bisher vorenthaltenen Versgenerierungsteil für alle vier Verstypen:

```
3 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel : SCRUDU - Gedichtgenerierung
5 :
6       MODE 2
7       BORDER 9,9
8       WIDTH 60
10 :
30      PAPER 2
40      PEN 1
50      CLS
60 :
100 REM ************ Supervisor ************
102 :
105         GOSUB 30000
110            REM Initialisation
160         GOSUB 40000
170            REM Titel
180 :
200        GOSUB 1000
210            REM Verstyp ermitteln
220 :
300       IF ty=1 THEN GOSUB 2000
310               REM Typ 1
400       IF ty=2 THEN GOSUB 3000
410               REM Typ 2
500       IF ty=3 THEN GOSUB 4000
510               REM Typ 3
600       IF ty=4 THEN GOSUB 5000
610               REM Typ 4
620 :
700        GOSUB 50000
710            REM weiterer Vers
720 GOTO 200
960 :
1000 REM ********* Verstyp ermitteln *******
1050       ty=1+INT(RND(1)*4)
1070 :
```

```
1080         IF ta=ty THEN GOTO 1000
1100         ta=ty
1200 :
1205       PRINT
1207       PRINT TAB (50);
1210       PAPER 0: PEN 1: PRINT "(Verstyp___";ty;"__)"
1215       PAPER 1: PEN 0
1220 :
1400 RETURN
1500 :
2000 REM ********* Verstyp Nr. 1 ***********
2003   GOSUB 23000
2005 :
2007 REM
2010 REM --------- erster Satz -------------
2015 :
2025 GOSUB 18000
2040      REM Nomen
2050   GOSUB 25000
2060      REM plus Praeposition
2070   GOSUB 21000
2080      REM plus Artikel
2090   GOSUB 23000
2100      REM plus Nomen
2140 :
2170 GOSUB 19000
2180   PRINT sa$
2195 :
2200 REM --------- zweiter Satz ------------
2210 :
2215 GOSUB 18000
2220    GOSUB 21000
2225       REM plus Artikel
2230    GOSUB 22000
2240       REM Adjektiv
2250    GOSUB 23000
2260       REM plus Nomen
2270 :
2275 GOSUB 19000
2390     PRINT sa$
2395 :
2400 REM --------- dritter Satz ------------
2420 :
2425 GOSUB 18000
2430   GOSUB 21000
2440     REM Artikel
2450   GOSUB 22000
2460     REM plus Adjektiv
2470   GOSUB 23000
2490     REM plus Nomen
2540 :
2570 GOSUB 19000
2590     PRINT sa$
2595 :
2600 RETURN
2610 :
```

```
3000 REM ********* Verstyp Nr. 2 ***********
3005 :
3007 REM
3010 REM --------- erster Satz -------------
3015 :
3025 GOSUB 18000
3030   GOSUB 21000
3040     REM Artikel
3070   GOSUB 22000
3080     REM plus Adjektiv
3110   GOSUB 23000
3120     REM plus Nomen
3140 :
3170 GOSUB 19000
3190     PRINT sa$
3195 :
3200 REM --------- zweiter Satz ------------
3210 :
3225 GOSUB 18000
3230   GOSUB 21000
3240     REM Artikel
3244   GOSUB 23000
3246     REM plus Nomen
3250   GOSUB 24000
3260     REM plus Verb
3270   GOSUB 25000
3280     REM plus Praeposition
3290   GOSUB 21000
3300     REM plus Artikel
3310   GOSUB 23000
3320     REM plus Nomen
3340 :
3370 GOSUB 19000
3380   PRINT sa$
3395 :
3400 REM --------- dritter Satz ------------
3420 :
3425 GOSUB 18000
3430   GOSUB 21000
3440     REM Artikel
3470   GOSUB 22000
3480     REM plus Adjektiv
3490   GOSUB 23000
3500     REM plus Nomen
3510   GOSUB 24000
3520     REM plus Verb
3540 :
3570 GOSUB 19000
3590    PRINT sa$
3595 :
3600 RETURN
3610 :
4000 REM ********* Verstyp Nr. 3 ***********
4005 :
4007 REM
4010 REM --------- erster Satz -------------
```

```
4015 :
4025 GOSUB 18000
4030   GOSUB 21000
4040     REM Artikel
4050   GOSUB 22000
4060     REM plus Adjektiv
4070   GOSUB 23000
4080     REM plus Nomen
4090   GOSUB 24000
4100     REM plus Verb
4140 :
4170 GOSUB 19000
4190    PRINT sa$
4195 :
4200 REM --------- zweiter Satz ------------
4210 :
4220  t3=77
4225 GOSUB 18000
4230   GOSUB 21000
4240     REM Artikel
4250   GOSUB 22000
4260     REM plus Adjektiv
4270   GOSUB 22000
4280     REM plus Adjektiv 2
4290   GOSUB 23000
4300     REM plus Nomen
4340 :
4370 GOSUB 19000
4390    PRINT sa$
4395 :
4400 REM --------- dritter Satz ------------
4420 :
4425 GOSUB 18000
4430   GOSUB 25000
4440     REM Praeposition
4450   GOSUB 21000
4460     REM plus Artikel
4470   GOSUB 22000
4480     REM plus Adjektiv
4490   GOSUB 23000
4500     REM plus Nomen
4540 :
4570 GOSUB 19000
4590    PRINT sa$
4595 :
4600 RETURN
4610 :
5000 REM ********* Verstyp Nr. 4 ***********
5005 :
5007 REM
5010 REM --------- erster Satz -------------
5015 :
5025 GOSUB 18000
5030   GOSUB 21000
5040     REM Artikel
5050   GOSUB 22000
```

```
5060      REM plus Adjektiv
5090    GOSUB 23000
5100      REM plus Nomen
5140 :
5170 GOSUB 19000
5190     PRINT sa$
5195 :
5200 REM --------- zweiter Satz ------------
5210 :
5225 GOSUB 18000
5230    GOSUB 25000
5240      REM Praeposition
5250    GOSUB 21000
5260      REM Artikel
5270    GOSUB 22000
5280      REM plus Adjektiv
5290    GOSUB 23000
5300      REM plus Nomen
5340 :
5370 GOSUB 19000
5390     PRINT sa$
5395 :
5400 REM --------- dritter Satz ------------
5420 :
5425 GOSUB 18000
5430  GOSUB 21000
5440     REM Artikel
5450  GOSUB 23000
5460     REM plus Nomen
5470  GOSUB 24000
5480     REM plus Verb
5540 :
5570 GOSUB 19000
5580     PRINT sa$
5595 :
5600 RETURN
5990 :
18000 REM ********* Satzfeld reset ***********
18050 n=0
18070 pa=0
18080 ar=0
18090 ad=0
18100     FOR i=0 TO 10
18110           w$(i)=""
18120     NEXT i
18130 RETURN
18990 :
19000 REM ********* Satzzuweisung  ***********
19080 sa$="  ": REM zwei Leerzeichen!
19100  FOR i= 1 TO n
19110     sa$=sa$+w$(i)+" "
19120  NEXT i
19200 RETURN
19990 :
20000 REM * Woerter aus Vokabular bestimmen **
20990 :
```

```
21000 REM ------------- Artikel --------------
21050  n=n+1
21070  ar=n
21100    vo=1+INT((RND)*4)
21200    w$(n)=a$(vo)
21900 RETURN
21990 :
22000 REM ------------ Adjektive -------------
22050  n=n+1
22060  ad=n
22100    vo=1+INT(RND(1)*50)
22200    w$(n)=ad$(vo)
22900 RETURN
22910 :
23000 REM -------------- Nomen ---------------
23050  n=n+1
23100    vo=1+INT(RND(1)*50)
23200    w$(n)=n$(vo)
23900 RETURN
23940 :
24000 REM -------------- Verben --------------
24050  n=n+1
24100    vo=1+INT(RND(1)*14)
24200    w$(n)=v$(vo)
24900 RETURN
24990 :
25000 REM ---------- Praepositionen ----------
25050  n=n+1
25070  pa=99
25100    vo=1+INT(RND(1)*6)
25200    w$(n)=p$(vo)
25900 RETURN
26000 :
30000 REM ********* Initialisation ***********
30010 :
30100 REM -------- Felder definieren ---------
30110  DIM a$(4)
30120     REM Artikel
30130  DIM ad$(50)
30140     REM Adjektive
30150  DIM n$(50)
30160     REM Nomen
30170  DIM v$(14)
30180     REM Verben
30190  DIM p$(6)
30200     REM Praeposition
30210 :
31000 REM ---------- Felder einlesen ---------
31010    REM Artikel
31020       FOR i=1 TO 4
31030          READ a$(i)
31040       NEXT i
31050 :
31100    REM Adjektive
31120       FOR i=1 TO 50
31130          READ ad$(i)
```

```
31140        NEXT i
31150 :
31200     REM Nomen
31220        FOR i=1 TO 50
31230           READ n$(i)
31240        NEXT i
31250 :
31300     REM Verben
31320        FOR i=1 TO 14
31330           READ v$(i)
31340        NEXT i
31360 :
31400     REM Praeposition
31420        FOR i=1 TO 6
31430           READ p$(i)
31440        NEXT i
31450 :
31500 RETURN
31510 :
32000 REM ------------- Daten ----------------
32010   REM Artikel -------------------------
32020     DATA der,das
32030     DATA ein,der
32080 :
33000   REM Adjektive -----------------------
33010     DATA junge,schwarze
33020     DATA wilde,verdorrte
33030     DATA umherschweifende,zinnoberrote
33040     DATA pochende,funkelnde
33050     DATA kleine,ruhige
33060     DATA rauhe,rote
33070     DATA alte,truebe
33080     DATA fruehe,schmale
33090     DATA frostige,gruene
33100     DATA spaete,lange
33110     DATA klare,sehnsuechtige
33120     DATA fallende,blaue
33130     DATA wiegende,feuchte
33140     DATA kalte,zerbrochene
33150     DATA klingende,gefleckte
33160     DATA feine,stille
33170     DATA verschneite,glaenzende
33180     DATA trockene,dunkle
33190     DATA schummrige,durchsichtige
33200     DATA neblige,leere
33210     DATA eisige,heisse
33220     DATA wolkige,warme
33230     DATA rauschende,toenende
33240     DATA herbstliche,reizende
33250     DATA beraubende,gelbe
33990 :
33995    REM Nomen -------------------------
34000     DATA Blume,Leuchtkaefer
34010     DATA Ruhe,Baum
34020     DATA Glitzern,Daemmerung
34030     DATA Funkeln,Sonne
```

```
34040      DATA See, Schneeflocke
34050      DATA Lichtung,Vogel
34060      DATA Feder,Dunst
34070      DATA Bach,Wald
34080      DATA Schatten,Nacht
34090      DATA Himmel,Brandung
34100      DATA Grass,Berg
34110      DATA Blume,Wasser
34120      DATA Geraeusch,Form
34130      DATA Feld,Tanne
34140      DATA Violett,Welle
34150      DATA Tau,Dunst
34160      DATA Blatt,Schatten
34170      DATA Schmetterling,Busch
34180      DATA Huegel,Wolke
34190      DATA Sonnenroete,Regen
34200      DATA Wind,Morgen
34210      DATA Meer,Schnee
34220      DATA Teich,Wasserfall
34230      DATA Brise,Fluss
34240      DATA Sonnenaufgang,Mond
34250 :
35000 REM Verben -----------------------------
35010      DATA ist gefallen,schleicht
35020      DATA faellt,ist getropft
35030      DATA trudelt,fliesst
35040      DATA raunt,schlaeft
35050      DATA schuettelt,treibt
35060      DATA hat gestoppt,kaempft
35070      DATA flattert,ist gestiegen
35080 :
36000 REM Praepositionen --------------------
36010      DATA unter,ueber
36020      DATA nahe,an
36030      DATA in,auf
38000 :
40000 REM ************* Titel ****************
40010 CLS
40020 WINDOW 1,70,1,5:PAPER 1:PEN 0:CLS
40030   PRINT: PRINT TAB(12)"S C R U D U"
40070   PRINT TAB(30)"(c) Olaf Hartwig"
40080   PRINT TAB(55)"1985"
40100 :
41000 REM ----- Gedichtstitel waehlen ------
41100 GOSUB 18000
41110   GOSUB 21000
41120      REM Artikel
41130   GOSUB 22000
41140      REM plus Adjektiv
41150   GOSUB 23000
41160      REM plus Nomen
41180 :
41200 GOSUB 19000
41250 WINDOW 3,60,7,7:CLS
41300    PRINT "  Titel___>  ";sa$
41500 :
```

```
41510        PAPER 1
41520        PEN 0
41530 WINDOW 5,79,9,25
41540        CLS
42000 RETURN
42120 :
50000 REM *** Taste=weiterer Vers,"s"=Stop ***
50005 :
50010   KE$=INKEY$:IF KE$="" THEN 50010
50020   IF KE$="s" THEN 980
50030 RETURN
```

3.2.2 Modifikation des Generatorprogramms

SCRUDU ist in der vorgestellten Version noch nicht ganz komplett. Das wird sehr schnell deutlich, wenn Sie sich einmal einen Beispiellauf anschauen. Ein typisches abstraktes Gedicht lautet:

die eisige See
der feine Welle
unter der frühe Mond
der Dunst flattert

der eisige Meer ist gestiegen
das rauhe pochende See
an ein stille Dunst

Was wir gerne als Ergebnis von SCRUDU hätten, sind die folgenden grammatikalisch korrekten Verse:

Die eisige See
die feine Welle
unter dem frühen Mond
der Dunst flattert

das eisige Meer ist gestiegen
die rauhe pochende See
an einem stillen Dunst

Ganz offensichtlich muß SCRUDU noch die Regeln des korrekten grammatischen Formates beherrschen lernen. Mit dieser Modifikation befassen wir uns im folgenden Abschnitt.

3.2.3 Das grammatische Format

Ändern Sie als erste Modifikation in Richtung eines korrekten grammatischen Formats zunächst den Nomenteil der Initialisierungs-Routine. Jedem Nomen wird ein Marker zugefügt. Er teilt SCRUDU mit, welche Deklination in Zusammenhang mit diesem Wort angewendet werden muß.

Die neue Initialisations-Routine lautet:

```
33995     REM Nomen -------------------------
34000      DATA Blume+,Leuchtkaefer-
34010      DATA Ruhe+,Baum-
34020      DATA Glitzern/,Daemmerung+
34030      DATA Funkeln/,Sonne+
34040      DATA See-, Schneeflocke+
34050      DATA Lichtung+,Vogel-
34060      DATA Feder+,Dunst-
34070      DATA Bach-,Wald-
34080      DATA Schatten-,Nacht+
34090      DATA Himmel-,Brandung+
34100      DATA Grass/,Berg-
34110      DATA Blume+,Wasser/
34120      DATA Geraeusch/,Form+
34130      DATA Feld/,Tanne+
34140      DATA Violett/,Welle+
34150      DATA Tau-,Dunst-
34160      DATA Blatt/,Schatten-
34170      DATA Schmetterling-,Busch-
34180      DATA Huegel-,Wolke+
34190      DATA Sonnenroete+,Regen-
34200      DATA Wind-,Morgen-
34210      DATA Meer/,Schnee-
34220      DATA Teich-,Wasserfall-
34230      DATA Brise+,Fluss-
34240      DATA Sonnenaufgang-,Mond-
```

Was bedeuten die einzelnen Marker?

Das Zeichen »–« veranlaßt SCRUDU später, dem mit diesem Marker gekennzeichneten Nomen nicht mehr wahllos alle beliebigen Artikel zuzuordnen. Statt dessen sind jetzt nur noch die beiden Artikel »der« oder »ein« möglich.

Der Marker »+« ermöglicht die Zuweisung der Artikel »die« oder »eine«.

Das Zeichen »/« bewirkt die Wahl des Artikels »das« oder »ein«, wobei das vorher markierte Adjektiv des Satzes die Endung »s« angehängt bekommt.

Zusätzlich zur Initialisierungs-Routine müssen wir nun noch eine Deklarations-Routine aufbauen, die diese Regeln anwendet. Sie wird aufgerufen durch die zusätzliche Programmzeile:

```
23250 gosub 27000
```

Fügen Sie anschließend die eigentliche Routine in das Programm ein:

```
27000 REM ****** Artikel korrekt zuweisen ****
27010   ko$=RIGHT$(w$(n),1)
27015   ar$=w$(ar)
27018    zu=1+INT(RND(1)*2)
27020    IF ko$="-" AND zu=1 THEN w$(ar)="der"
27025    IF ko$="-" AND pa=99 THEN w$(ar)="ein":w$(ad)=
         w$(ad)+"n":GOTO 28000
27030    IF ko$="-" AND zu=2 THEN w$(ar)="ein":w$(ad)=w$(ad)+"r"
```

```
27100    IF ko$="+" AND zu=1 THEN w$(ar)="die"
27110    IF ko$="+" AND zu=2 THEN w$(ar)="eine"
27210    IF ko$="/" AND zu=1 THEN w$(ar)="ein":w$(ad)=w$(ad)+"s"
27220    IF ko$="/" AND zu=2 THEN w$(ar)="das"
27300 :
27500    IF t3=77 THEN t3=0: ad=ad-1: GOTO 27020
27510       REM zwei  Adjektive  vorhanden,  daher
27520       REM Wiederholung fuer voriges adjektiv.
27600 :
28000  w$(n)=LEFT$(w$(n),LEN(w$(n))-1)
28010    REM Marker fuer Geschlecht entfernen (/,-.+)
28050 :
28100 RETURN
```

Mit den vorangegangenen Modifikationen ist SCRUDU komplett. Damit Sie bei der Eingabe des recht umfangreichen Programms keine Probleme mit der Reihenfolge der einzelnen Module bekommen, finden Sie auf den folgenden Seiten das komplette Listing des optimierten SCRUDU-Programms:

```
3 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel : SCRUDU - Gedichtgenerierung
5 :
6       MODE 2
7       BORDER 9,9
8       WIDTH 60
10 :
30      PAPER 2
40      PEN 1
50      CLS
70 :
100 REM ************ Supervisor ************
102 :
105         GOSUB 30000
110            REM Initialisation
160         GOSUB 40000
170            REM Titel
180 :
200        GOSUB 1000
210            REM Verstyp ermitteln
220 :
300       IF ty=1 THEN GOSUB 2000
310               REM Typ 1
400       IF ty=2 THEN GOSUB 3000
410               REM Typ 2
500       IF ty=3 THEN GOSUB 4000
510               REM Typ 3
600       IF ty=4 THEN GOSUB 5000
610               REM Typ 4
620 :
700        GOSUB 50000
710            REM weiterer Vers
720 GOTO 200
960 :
980 :
```

```
990 :
1000 REM ********* Verstyp ermitteln *******
1050      ty=1+INT(RND(1)*4)
1070 :
1080        IF ta=ty THEN GOTO 1000
1100        ta=ty
1200 :
1205      PRINT
1207      PRINT TAB (50);
1210      PAPER 0: PEN 1: PRINT "(Verstyp___";ty;"__)"
1215      PAPER 1: PEN 0
1220 :
1400 RETURN
1500 :
1510 :
2000 REM ********* Verstyp Nr. 1 ***********
2003   GOSUB 23000
2005 :
2007 REM
2010 REM --------- erster Satz -------------
2015 :
2025 GOSUB 18000
2040      REM Nomen
2050   GOSUB 25000
2060      REM plus Praeposition
2070   GOSUB 21000
2080      REM plus Artikel
2090   GOSUB 23000
2100      REM plus Nomen
2140 :
2160 :
2170 GOSUB 19000
2180   PRINT sa$
2195 :
2200 REM --------- zweiter Satz ------------
2210 :
2215 GOSUB 18000
2220    GOSUB 21000
2225       REM plus Artikel
2230    GOSUB 22000
2240       REM Adjektiv
2250    GOSUB 23000
2260       REM plus Nomen
2270 :
2275 GOSUB 19000
2390     PRINT sa$
2395 :
2400 REM --------- dritter Satz ------------
2420 :
2425 GOSUB 18000
2430   GOSUB 21000
2440     REM Artikel
2450   GOSUB 22000
2460     REM plus Adjektiv
2470   GOSUB 23000
2490     REM plus Nomen
```

```
2540 :
2560 :
2570 GOSUB 19000
2590     PRINT sa$
2595 :
2600 RETURN
2610 :
2990 :
3000 REM ********* Verstyp Nr. 2 ***********
3005 :
3007 REM
3010 REM --------- erster Satz -------------
3015 :
3025 GOSUB 18000
3030   GOSUB 21000
3040     REM Artikel
3070   GOSUB 22000
3080     REM plus Adjektiv
3110   GOSUB 23000
3120     REM plus Nomen
3140 :
3150 :
3170 GOSUB 19000
3190     PRINT sa$
3195 :
3200 REM --------- zweiter Satz ------------
3210 :
3225 GOSUB 18000
3230   GOSUB 21000
3240     REM Artikel
3244   GOSUB 23000
3246     REM plus Nomen
3250   GOSUB 24000
3260     REM plus Verb
3270   GOSUB 25000
3280     REM plus Praeposition
3290   GOSUB 21000
3300     REM plus Artikel
3310   GOSUB 23000
3320     REM plus Nomen
3340 :
3360 :
3370 GOSUB 19000
3380   PRINT sa$
3395 :
3400 REM --------- dritter Satz ------------
3420 :
3425 GOSUB 18000
3430   GOSUB 21000
3440     REM Artikel
3470   GOSUB 22000
3480     REM plus Adjektiv
3490   GOSUB 23000
3500     REM plus Nomen
3510   GOSUB 24000
3520     REM plus Verb
```

```
3540 :
3550 :
3570 GOSUB 19000
3590    PRINT sa$
3595 :
3600 RETURN
3610 :
3990 :
4000 REM ********* Verstyp Nr. 3 ***********
4005 :
4007 REM
4010 REM --------- erster Satz -------------
4015 :
4025 GOSUB 18000
4030   GOSUB 21000
4040     REM Artikel
4050   GOSUB 22000
4060     REM plus Adjektiv
4070   GOSUB 23000
4080     REM plus Nomen
4090   GOSUB 24000
4100     REM plus Verb
4140 :
4150 :
4170 GOSUB 19000
4190    PRINT sa$
4195 :
4200 REM --------- zweiter Satz ------------
4210 :
4220  t3=77
4225 GOSUB 18000
4230   GOSUB 21000
4240     REM Artikel
4250   GOSUB 22000
4260     REM plus Adjektiv
4270   GOSUB 22000
4280     REM plus Adjektiv 2
4290   GOSUB 23000
4300     REM plus Nomen
4340 :
4360 :
4370 GOSUB 19000
4390    PRINT sa$
4395 :
4400 REM --------- dritter Satz ------------
4420 :
4425 GOSUB 18000
4430   GOSUB 25000
4440     REM Praeposition
4450   GOSUB 21000
4460     REM plus Artikel
4470   GOSUB 22000
4480     REM plus Adjektiv
4490   GOSUB 23000
4500     REM plus Nomen
4540 :
```

```
4560 :
4570 GOSUB 19000
4590    PRINT sa$
4595 :
4600 RETURN
4610 :
4620 :
5000 REM ********* Verstyp Nr. 4 ***********
5005 :
5007 REM
5010 REM --------- erster Satz -------------
5015 :
5025 GOSUB 18000
5030   GOSUB 21000
5040     REM Artikel
5050   GOSUB 22000
5060     REM plus Adjektiv
5090   GOSUB 23000
5100     REM plus Nomen
5140 :
5150 :
5170 GOSUB 19000
5190    PRINT sa$
5195 :
5200 REM --------- zweiter Satz ------------
5210 :
5225 GOSUB 18000
5230   GOSUB 25000
5240     REM Praeposition
5250   GOSUB 21000
5260     REM Artikel
5270   GOSUB 22000
5280     REM plus Adjektiv
5290   GOSUB 23000
5300     REM plus Nomen
5340 :
5350 :
5370 GOSUB 19000
5390    PRINT sa$
5395 :
5400 REM --------- dritter Satz ------------
5420 :
5425 GOSUB 18000
5430  GOSUB 21000
5440    REM Artikel
5450  GOSUB 23000
5460    REM plus Nomen
5470  GOSUB 24000
5480    REM plus Verb
5540 :
5550 :
5570 GOSUB 19000
5580    PRINT sa$
5595 :
5600 RETURN
5990 :
```

```
5995 :
18000 REM ********* Satzfeld reset ***********
18050 n=0
18070 pa=0
18080 ar=0
18090 ad=0
18100    FOR i=0 TO 10
18110         w$(i)=""
18120    NEXT i
18130 RETURN
18990 :
19000 REM ********* Satzzuweisung  ***********
19080 sa$="  ": REM zwei Leerzeichen!
19100  FOR i= 1 TO n
19110     sa$=sa$+w$(i)+"  "
19120  NEXT i
19200 RETURN
19990 :
20000 REM * Woerter aus Vokabular bestimmen **
20990 :
21000 REM ------------- Artikel -------------
21050  n=n+1
21070  ar=n
21100    vo=1+INT((RND)*4)
21200    w$(n)=a$(vo)
21900 RETURN
21990 :
22000 REM ------------ Adjektive ------------
22050  n=n+1
22060  ad=n
22100    vo=1+INT(RND(1)*50)
22200    w$(n)=ad$(vo)
22900 RETURN
22910 :
23000 REM -------------- Nomen ---------------
23050  n=n+1
23100    vo=1+INT(RND(1)*50)
23200    w$(n)=n$(vo)
23250  GOSUB 27000
23900 RETURN
23940 :
24000 REM -------------- Verben --------------
24050  n=n+1
24100    vo=1+INT(RND(1)*14)
24200    w$(n)=v$(vo)
24900 RETURN
24990 :
25000 REM ---------- Praepositionen ----------
25050  n=n+1
25070  pa=99
25100    vo=1+INT(RND(1)*6)
25200    w$(n)=p$(vo)
25900 RETURN
25990 :
26000 :
```

```
27000 REM ****** Artikel korrekt zuweisen ****
27010   ko$=RIGHT$(w$(n),1)
27015   ar$=w$(ar)
27018    zu=1+INT(RND(1)*2)
27020    IF ko$="-" AND zu=1 THEN w$(ar)="der"
27025    IF ko$="-" AND pa=99 THEN w$(ar)="ein":w$(ad)=
         w$(ad)+"n":GOTO 28000
27030    IF ko$="-" AND zu=2 THEN w$(ar)="ein":w$(ad)=w$(ad)+"r"
27100    IF ko$="+" AND zu=1 THEN w$(ar)="die"
27110    IF ko$="+" AND zu=2 THEN w$(ar)="eine"
27210    IF ko$="/" AND zu=1 THEN w$(ar)="ein":w$(ad)=w$(ad)+"s"
27220    IF ko$="/" AND zu=2 THEN w$(ar)="das"
27300 :
27500    IF t3=77 THEN t3=0: ad=ad-1: GOTO 27020
27510       REM zwei  Adjektive  vorhanden,  daher
27520       REM Wiederholung fuer voriges adjektiv.
27600 :
28000  w$(n)=LEFT$(w$(n),LEN(w$(n))-1)
28010    REM Marker fuer Geschlecht entfernen (/,-.+)
28050 :
28100 RETURN
29980 :
29990 :
30000 REM ********* Initialisation ***********
30010 :
30100 REM -------- Felder definieren ---------
30110  DIM a$(4)
30120     REM Artikel
30130  DIM ad$(50)
30140     REM Adjektive
30150  DIM n$(50)
30160     REM Nomen
30170  DIM v$(14)
30180     REM Verben
30190  DIM p$(6)
30200     REM Praeposition
30210 :
31000 REM ---------- Felder einlesen ---------
31010    REM Artikel
31020       FOR i=1 TO 4
31030          READ a$(i)
31040       NEXT i
31050 :
31100    REM Adjektive
31120       FOR i=1 TO 50
31130          READ ad$(i)
31140       NEXT i
31150 :
31200    REM Nomen
31220       FOR i=1 TO 50
31230          READ n$(i)
31240       NEXT i
31250 :
31300    REM Verben
31320       FOR i=1 TO 14
31330          READ v$(i)
```

```
31340       NEXT i
31360 :
31400    REM Praeposition
31420       FOR i=1 TO 6
31430          READ p$(i)
31440       NEXT i
31450 :
31500 RETURN
31510 :
32000 REM ------------- Daten ----------------
32010   REM Artikel -------------------------
32020     DATA der,das
32030     DATA ein,der
32070 :
32080 :
33000   REM Adjektive -----------------------
33010     DATA junge,schwarze
33020     DATA wilde,verdorrte
33030     DATA umherschweifende,zinnoberrote
33040     DATA pochende,funkelnde
33050     DATA kleine,ruhige
33060     DATA rauhe,rote
33070     DATA alte,truebe
33080     DATA fruehe,schmale
33090     DATA frostige,gruene
33100     DATA spaete,lange
33110     DATA klare,sehnsuechtige
33120     DATA fallende,blaue
33130     DATA wiegende,feuchte
33140     DATA kalte,zerbrochene
33150     DATA klingende,gefleckte
33160     DATA feine,stille
33170     DATA verschneite,glaenzende
33180     DATA trockene,dunkle
33190     DATA schummrige,durchsichtige
33200     DATA neblige, leere
33210     DATA eisige,heisse
33220     DATA wolkige,warme
33230     DATA rauschende,toenende
33240     DATA herbstliche,reizende
33250     DATA beraubende,gelbe
33990 :
33995    REM Nomen -------------------------
34000     DATA Blume+,Leuchtkaefer-
34010     DATA Ruhe+,Baum-
34020     DATA Glitzern/,Daemmerung+
34030     DATA Funkeln/,Sonne+
34040     DATA See-, Schneeflocke+
34050     DATA Lichtung+,Vogel-
34060     DATA Feder+,Dunst-
34070     DATA Bach-,Wald-
34080     DATA Schatten-,Nacht+
34090     DATA Himmel-,Brandung+
34100     DATA Grass/,Berg-
34110     DATA Blume+,Wasser/
34120     DATA Geraeusch/,Form+
```

```
34130     DATA Feld/,Tanne+
34140     DATA Violett/,Welle+
34150     DATA Tau-,Dunst-
34160     DATA Blatt/,Schatten-
34170     DATA Schmetterling-,Busch-
34180     DATA Huegel-,Wolke+
34190     DATA Sonnenroete+,Regen-
34200     DATA Wind-,Morgen-
34210     DATA Meer/,Schnee-
34220     DATA Teich-,Wasserfall-
34230     DATA Brise+,Fluss-
34240     DATA Sonnenaufgang-,Mond-
34250 :
35000 REM Verben ----------------------------
35010     DATA ist gefallen,schleicht
35020     DATA faellt,ist getropft
35030     DATA trudelt,fliesst
35040     DATA raunt,schlaeft
35050     DATA schuettelt,treibt
35060     DATA hat gestoppt,kaempft
35070     DATA flattert,ist gestiegen
35080 :
36000 REM Praepositionen ---------------------
36010     DATA unter,ueber
36020     DATA nahe,an
36030     DATA in,auf
37000 :
38000 :
40000 REM ************* Titel ****************
40010 CLS
40020 WINDOW 1,70,1,5:PAPER 1:PEN 0:CLS
40030   PRINT: PRINT TAB(12)"S C R U D U"
40070   PRINT TAB(30)"(c) Olaf Hartwig"
40080   PRINT TAB(55)"1985"
40100 :
41000 REM ----- Gedichtstitel waehlen ------
41100 GOSUB 18000
41110   GOSUB 21000
41120      REM Artikel
41130   GOSUB 22000
41140      REM plus Adjektiv
41150   GOSUB 23000
41160      REM plus Nomen
41170 :
41180 :
41200 GOSUB 19000
41250 WINDOW 3,60,7,7:CLS
41300    PRINT "  Titel___>  ";sa$
41400 :
41500 :
41510       PAPER 1
41520       PEN 0
41530 WINDOW 5,79,9,25
41540       CLS
42000 RETURN
42110 :
```

```
42120 :
50000 REM *** Taste=weiterer Vers,"s"=Stop ***
50005 :
50010   KE$=INKEY$:IF KE$="" THEN 50010
50020   IF KE$="s" THEN 980
50030 RETURN
```

Genießen Sie ruhig einmal die korrekte Dichtung eines Probelaufs. Sie können durch Änderung des Initialisationsteils den Sprachschatz von SCRUDU beliebig verändern und so neue Gedichtsthemen erzeugen.

Ein Demonstrationslauf von SCRUDU

Ein Beispielgedichtsteil lautet im nun korrekten grammatikalischen Format folgendermaßen:

Ein fallender Schatten
der beraubende Dunst raunt
der schwarze gefleckte Regen
nahe einem dunklen Mond
eine eisige Nacht fällt
eine ruhige frostige Dämmerung
an einen nebligen See

Auch hier möchte ich Sie gerne zu eigenen Experimenten auffordern. Als Anregung vielleicht ein interessanter Hinweis: KI-Experten der Stanford University haben kürzlich ein Programm entwickelt, das selbständig Drehbücher zu den TV-Serien »Dallas« und »Denver« schreibt. Das Grundprinzip des Programms entspricht dem von SCRUDU, nur daß die neu aneinandergefügten Sprachelemente, also Wörter und einzelne Sätze, durch eine vom Computer vorbestimmte Rahmenhandlung kombiniert werden.
Ein derartiges Programm sprengt natürlich die Grenzen eines Home-Computers, einzelne Aspekte lassen sich jedoch übertragen. Hier eröffnet sich Ihnen ein weites Betätigungsfeld.

3.3 Grafische Computer-Kreativität

Ich möchte Sie in diesem Abschnitt in einen weiteren Teilbereich der Computer-Kreativität einführen.
Wir werden uns hier mit der Computer-Kunst befassen und wenden uns im speziellen dem Entwurf von Berglandschaften zu. Das Besondere ist dabei, daß die Landschaften nicht wie normalerweise üblich durch ein Draw-Programm von einem Menschen eingegeben und erstellt werden. Statt dessen werden sie von einem Programm automatisch erzeugt.

Erstaunlich ist dabei, daß keine der Landschaften einer anderen gleicht. Verblüffend ist aber auch, mit welch minimalem Aufwand sich die Landschaftsbilder erzeugen lassen:

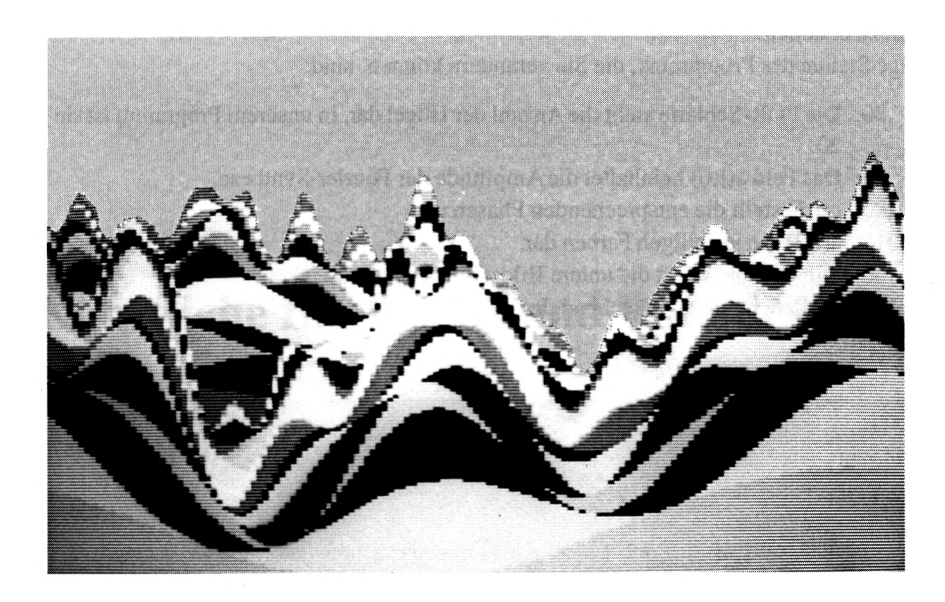


Bild 3.1

Das zum Generieren der Landschaftsbilder notwendige Programm sieht wie folgt aus:

```
10 REM COMPUTERKUNST+'Kreativitaet'
20 REM (c) Olaf Hartwig '85
25 '
30 DIM a(20,2)
40 MODE 0
60 INK 0,11
70 '
80  FOR i=0 TO 20
90     a(i,0)=RND*100/(i+1)
100    a(i,1)=RND*2*PI
110    a(i,2)=RND*15
120 NEXT i
130 '
140 FOR i=1 TO 640 STEP 4
150  w=2*i/640*PI
160  y=9
170   PLOT i-1,1
180    FOR p=0 TO 20
190      y=y+a(p,0)*(1+SIN(p*w+a(p,1)))
200      DRAW i-1,y,a(p,2)
210    NEXT p
220 NEXT i
```

Es handelt sich bei dem verwendeten Algorithmus um eine Mini-Version einer Fourier-Synthese. Das Programm wurde kompakt gehalten, um dadurch Ihre Experimentierfreudigkeit zu wecken. Es beinhaltet den kompletten Algorithmus und erlaubt es, eindrucksvolle Berglandschaften ganz spontan zu erstellen.

Wichtige Stellen des Programms, die Sie verändern können, sind:

Zeile 80: Die FOR-Schleife stellt die Anzahl der Hügel dar, in unserem Programm ist sie gleich 20.

Zeile 90: Das Feld a(i,0) beinhaltet die Amplitude der Fourier-Synthese.

Zeile 100: a(i,1) stellt die entsprechenden Phasen und

Zeile 110: a(i,2) die jeweiligen Farben dar.

Zeile 160: Die Variable y legt die untere Bildschirmgrenze fest.

Zeile 180: FOR-Schleife entsprechend der Zeile 80.

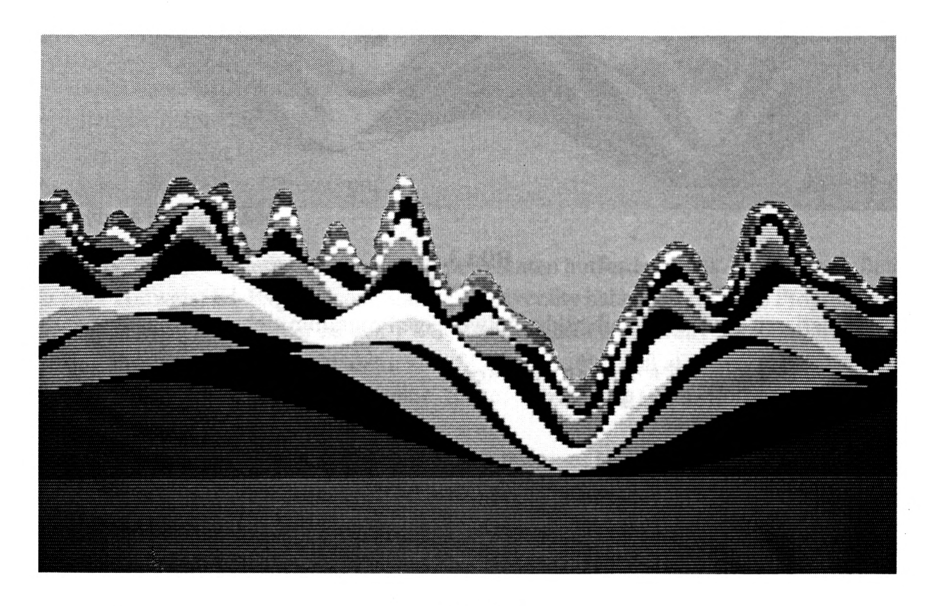


Bild 3.2

4 Künstliche Intelligenz und Robotics

4.1 Der Micro-Mouse-Wettbewerb

In diesem Kapitel werden wir ein Spezialgebiet der KI, Robotics, anhand eines konkreten Beispiels, des MICRO-MOUSE MAZE CONTEST, untersuchen.
Dieser jedes Jahr stattfindende Wettbewerb ist ein gutes Beispiel für die Verknüpfung von Künstlicher Intelligenz und Robotics. Worum geht es? Eine Micro-Mouse ist natürlich keine richtige Maus, sondern ein Miniatur-Roboter, der von einem eingebauten Mikroprozessor gesteuert wird. Die kleinen und intelligenten Maschinen suchen sich selber ihren Weg durch ein Labyrinth.
Der zu bewältigende Irrgarten, in dem ein Ziel gefunden werden muß, ist ihnen dabei unbekannt. Jede Maus bekommt im Wettbewerb 15 Minuten Zeit, um die Struktur des Labyrinths zu erlernen.
Im wesentlichen hängt der Erfolg einer Maus von zwei Faktoren ab. Der entscheidende Faktor ist die Effizienz der Software. Mit diesem Bereich werden wir uns gleich intensiver beschäftigen. Da der Wettbewerb gegen die Uhr ausgetragen wird, spielt der mechanische Aufbau der Maus ebenfalls eine wichtige Rolle. Je flinker sie ist, je wendiger und schneller, desto besser sind ihre Erfolgsaussichten.
Wozu dieser Wettbewerb? Ist das eigentlich nur Spielerei? Weit gefehlt, die Idee hat einen ganz ernsthaften Hintergrund. Auf spielerische Weise werden nützliche Erkenntnisse zur Verbesserung von Industrie-Robotern gewonnen.
Wir wollen uns nun damit beschäftigen, wie eine Maus so programmiert werden kann, daß sie selbst ihren Weg aus einem Labyrinth findet, und dazu drei verschiedene Lösungsalgorithmen untersuchen.

4.2 Der MMC-BRAIN-Simulations-Algorithmus

Dieser Lösungsalgorithmus funktioniert nach folgendem Grundprinzip: zunächst wird das Labyrinth in ein Datenfeld eingelesen. Anschließend bewegt sich die Maus so lange in einer Richtung durch den Irrgarten, bis sie auf eine Wand trifft. Dann wird die Marschrichtung geändert und der Kurs wird fortgesetzt.

Damit sich die Maus nicht in einer Endlosschleife verfangen kann, also denselben Weg immer wiederholt, wird die bereits zurückgelegte Strecke markiert.

Der Supervisor

Der Programmkopf besteht aus nur drei Unterroutinen, der Initialisierung, dem eigentlichen Lösungsalgorithmus und dem Programmende, das angesprungen wird, wenn der Ausgang des Labyrinths entdeckt wurde:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: MMC-BRAIN Version 1
5 REM Micromouse Contest
6 :
7   BORDER 9,9
8   PAPER 0
9   PEN 1:CLS
10  BORDER 9,9
11  MODE 1
12  WIDTH 60
20 :
80 mar=7
85 :
90 REM Supervisor
100 GOSUB 1000
110     REM Initialisation
200 GOSUB 2000
210     REM Hauptprogramm
300 GOSUB 5000
310     REM Ausgang gefunden
950 :
960 STOP
```

Die Initialisierung

Die Prozedur liest den Irrgarten in das Feld »f$(x,y)« ein, legt die Startkoordinaten der Maus fest und setzt den Zähler der Durchgänge »s« auf Null. Zusätzlich wird der Aufbau des Bildschirms gestaltet und der Titel ausgegeben.

Das sieht so aus:

```
1000 REM Initialisation
1005 :
1010 CLS
1020 PRINT TAB(20)"MMC-BRAIN"
```

```
1030 PRINT
1040 PRINT TAB(20)"(c) Olaf Hartwig 1985"
1045 ORIGIN 300,1:DRAW 1,400
1047 ORIGIN 300,330:DRAW 340,1
1048 ORIGIN 1,110:DRAW 300,1
1050 :
1090  REM Labyrinth lesen
1100   DIM f$(17,15)
1110     FOR y=1 TO 16
1120       READ hi$
1130         FOR x=1 TO 15
1140           f$(y,x)=MID$(hi$,x,1)
1150         NEXT x
1160     NEXT y
1200  s=0
1210  REM Schleifenzaehler
1220 :
1280 RETURN
1290 :
1500 REM Labyrinth
1600 DATA ***************
1610 DATA * *   *  *    *
1620 DATA *   * **** ** *
1630 DATA * **       *  *
1640 DATA * *        *  *
1650 DATA * *           *
1660 DATA * ***** **** **
1670 DATA *       * *  **
1680 DATA *****  **** ***
1690 DATA *             *
1700 DATA * * *** ***** *
1710 DATA *     *       *
1720 DATA * ***   ** * **
1730 DATA * *      * *  *
1740 DATA *      *     **
1750 DATA **  ***********
```

Sie können das Labyrinth innerhalb des Rahmenmaßes von 15 Pixel in der Breite und 16 Pixel in der Höhe beliebig für eigene Experimente verändern.

Der Lösungsalgorithmus

Zunächst wird die Marschrichtung der Maus durch die Variable »e« bestimmt. Dabei bedeuten

e=1 Norden,
e=2 Osten,
e=3 Süden und
e=4 Westen.

Je nach Wert der Variablen »e« wird eine entsprechende Unterroutine für jede Richtung aufgerufen. Jede der Routinen bestimmt, ob der in dieser Richtung nächste Punkt ein Wandzeichen (*) oder ein Wegmarker (.) ist. Ist er ein Wegmarker, so wird die Richtungszuweisung wiederholt.

Der Programmteil lautet:

```
2000 REM Hauptprogramm
2005 :
2006 IF mar=7 THEN GOSUB 6000
2010 yalt=y:xalt=x
2020 :
2100 e=1+INT(RND(1)*4)
2103 mk=0
2104 REM Wandmarker reset
2105 :
2110 ON e GOSUB 2200,2300,2400,2500
2150 :
2160 IF mk=99 THEN 2010
2170 REM Wand ist im Weg
2180 :
2185 IF mk=11 THEN 2010
2187 REM Wiederholung des alten Wegs
2188 :
2190 GOTO 3000
2195 :
2200 REM Norden
2210  IF f$(y+1,x)="*" THEN mk=99:RETURN
2215  IF f$(y+1,x)="." THEN mk=11:RETURN
2220     GOSUB 2700
2240   y=y+1
2290 RETURN
2295 :
2300 REM Osten
2310  IF f$(y,x+1)="*" THEN mk=99:RETURN
2315  IF f$(y,x+1)="." THEN mk=11:RETURN
2320     GOSUB 2700
2340   x=x+1
2390 RETURN
2395 :
2400 REM Sueden
2410  IF f$(y-1,x)="*" THEN mk=99:RETURN
2415  IF f$(y-1,x)="." THEN mk=11:RETURN
2420     GOSUB 2700
2440  y=y-1
2490 RETURN
2495 :
2500 REM Westen
2510  IF f$(y,x-1)="*" THEN mk=99:RETURN
2515  IF f$(y,x-1)="." THEN mk=11:RETURN
2520     GOSUB 2700
2540  x=x-1
2590 RETURN
2595 :
2700 REM 'Gedaechnis' - Mausweg festhalten
2720   f$(y,x)="."
2740 RETURN
```

Die Zielabfrage

Als nächstes muß das Hauptprogramm überprüfen, ob es am Ziel angekommen ist, und wenn nicht, die Richtung der Maus sowie ihre aktuelle Position im Labyrinth ausgeben. Im Programm sieht das so aus:

```
3000 REM cont.
3010 s=s+1
3020 REM Durchgaenge zaehlen
3100 REM Am Ziel angelangt?
3110 IF y=16 THEN RETURN
3990 :
4000 REM Ausdruck des Labyrinths
4005 :
4010 zaehler=0
4050 LOCATE xalt,yalt:PRINT"."
4152 :
4153 PRINT
4154 LOCATE 1,20
4155 PRINT "Richtung..>";e
4157 PRINT
4159 IF e=1 THEN PRINT "   Sueden"
4161 IF e=2 THEN PRINT "   Osten "
4163 IF e=3 THEN PRINT "   Norden"
4165 IF e=4 THEN PRINT "   Westen"
4169 PRINT
4170 :
4172 LOCATE 1,24
4175 PRINT "Durchgang Nr.";s
4180 :
4200 REM Mausposition
4220  LOCATE x,y
4230  PRINT"M"
4240 :
4400 GOTO 2010
```

Die Labyrinthausgabe

Die Routine zur Ausgabe des kompletten Labyrinths befindet sich in den Zeilen 6000–6100 und wird von der Zeile 2006 aus aufgerufen.
Sie lautet:

```
6000 REM Ausdruck
6002 LOCATE 1,1
6005 FOR y=1 TO 16
6010   FOR x=1 TO 15
6020     PRINT f$(y,x);
6030   NEXT x
6040  PRINT
6050 NEXT y
6070 x=2:y=2
6080 REM Startposition der Maus
6090 mar=0
6100 RETURN
```

Die Endroutine

Das kurze Unterprogramm gibt eine Statusmeldung in einem Window aus und zeigt die Anzahl der benötigten Durchgänge der Maus, die zum Entdecken des Ausgangs benötigt wurden, an:

```
5000 REM Ausgang gefunden
5010 :
5020 WINDOW 20,39,12,19
5022 PAPER 1:PEN 0:CLS
5028 PRINT
5030 PRINT " Ausgang gefunden"
5035 PRINT
5040 PRINT s;" Durchgaenge."
5100 PRINT
5110 PRINT" Druecken Sie eine   Taste..."
5120 k$=INKEY$:IF k$="" THEN 5120
5200 RETURN
```

MMC-BRAIN-Modifikation

Wenn Sie nun das Programm laufen lassen, werden Sie schnell einen Schönheitsfehler feststellen. Die Maus kann sich durch die Markierung ihres bisherigen Weges zwar nicht mehr in einer Endlosschleife verfangen, doch ist es ihr möglich, sich ihren Weg zu »verbauen«, d.h. sich durch die Wegmarker einzuschließen.

Das Problem kann man durch einige Änderungen lösen. Unten sehen Sie das komplette MMC-BRAIN-Programmm mit allen nötigen Korrekturen. Die Hauptänderung befindet sich in den Zeilen 3500 ff.

Das neue MMC-BRAIN-Programm lautet komplett folgendermaßen:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: MMC-BRAIN Version 2
5 REM Micromouse Contest
6 :
7   BORDER 9,9
8   PAPER 0
9   PEN 1:CLS
10  BORDER 9,9
11  MODE 1
12  WIDTH 60
20 :
80 mar=7
85 :
90 REM Supervisor
100 GOSUB 1000
110    REM Initialisation
200 GOSUB 2000
210    REM Hauptprogramm
300 GOSUB 5000
310    REM Ausgang gefunden
950 :
960 STOP
970 :
```

```
980 REM Prozeduren
990 :
1000 REM Initialisation
1005 :
1010 CLS
1020 PRINT TAB(20)"MMC-BRAIN"
1030 PRINT
1040 PRINT TAB(20)"(c) Olaf Hartwig 1985"
1045 ORIGIN 300,1:DRAW 1,400
1047 ORIGIN 300,330:DRAW 340,1
1048 ORIGIN 1,110:DRAW 300,1
1050 :
1090  REM Labyrinth lesen
1100   DIM f$(17,15)
1110     FOR y=1 TO 16
1120       READ hi$
1130         FOR x=1 TO 15
1140           f$(y,x)=MID$(hi$,x,1)
1150         NEXT x
1160     NEXT y
1200  s=0
1210  REM Schleifenzaehler
1220 :
1280 RETURN
1290 :
1500 REM Labyrinth
1600 DATA ***************
1610 DATA * *   *  *    *
1620 DATA *   * **** ** *
1630 DATA * **       *  *
1640 DATA * *        *  *
1650 DATA * *           *
1660 DATA * ***** **** **
1670 DATA *       * *  **
1680 DATA *****  **** ***
1690 DATA *             *
1700 DATA * * *** ***** *
1710 DATA *     *       *
1720 DATA * ***   ** * **
1730 DATA * *      * *  *
1740 DATA *      *     **
1750 DATA **  ***********
1780 :
2000 REM Hauptprogramm
2005 :
2006 IF mar=7 THEN GOSUB 6000
2007 zaehler=zaehler+1
2008 IF zaehler>30 THEN 3500
2010 yalt=y:xalt=x
2020 :
2100 e=1+INT(RND(1)*4)
2103 mk=0
2104 REM Wandmarker reset
2105 :
2110 ON e GOSUB 2200,2300,2400,2500
2150 :
```

```
2160 IF mk=99 THEN 2007
2170 REM Wand ist im Weg
2180 :
2185 IF mk=11 THEN 2007
2187 REM Wiederholung des alten Wegs
2188 :
2190 GOTO 3000
2195 :
2200 REM Norden
2210  IF f$(y+1,x)="*" THEN mk=99:RETURN
2215  IF f$(y+1,x)="." THEN mk=11:RETURN
2220     GOSUB 2700
2240   y=y+1
2290 RETURN
2295 :
2300 REM Osten
2310  IF f$(y,x+1)="*" THEN mk=99:RETURN
2315  IF f$(y,x+1)="." THEN mk=11:RETURN
2320     GOSUB 2700
2340   x=x+1
2390 RETURN
2395 :
2400 REM Sueden
2410  IF f$(y-1,x)="*" THEN mk=99:RETURN
2415  IF f$(y-1,x)="." THEN mk=11:RETURN
2420     GOSUB 2700
2440  y=y-1
2490 RETURN
2495 :
2500 REM Westen
2510  IF f$(y,x-1)="*" THEN mk=99:RETURN
2515  IF f$(y,x-1)="." THEN mk=11:RETURN
2520     GOSUB 2700
2540  x=x-1
2590 RETURN
2595 :
2700 REM 'Gedaechnis' - Mausweg festhalten
2720   f$(y,x)="."
2740 RETURN
2800 :
3000 REM cont.
3010 s=s+1
3020 REM Durchgaenge zaehlen
3100 REM Am Ziel angelangt?
3110 IF y=16 THEN RETURN
3120 :
3200 GOTO 4000
3500 REM Reset wenn Weg verbaut ist
3510   FOR i=1 TO 16
3520     FOR k=1 TO 15
3530        IF f$(i,k)="." THEN f$(i,k)=" "
3540     NEXT k
3550   :
3560   NEXT i
3600 GOSUB 6000
3610 xalt=2:yalt=2
```

```
3620 s=0
3990 :
4000 REM Ausdruck des Labyrinths
4005 :
4010 zaehler=0
4050 LOCATE xalt,yalt:PRINT"."
4152 :
4153 PRINT
4154 LOCATE 1,20
4155 PRINT "Richtung..>";e
4157 PRINT
4159 IF e=1 THEN PRINT "   Sueden"
4161 IF e=2 THEN PRINT "   Osten "
4163 IF e=3 THEN PRINT "   Norden"
4165 IF e=4 THEN PRINT "   Westen"
4169 PRINT
4170 :
4172 LOCATE 1,24
4175 PRINT "Durchgang Nr.";s
4180 :
4200 REM Mausposition
4220  LOCATE x,y
4230  PRINT"M"
4240 :
4400 GOTO 2007
4440 :
5000 REM Ausgang gefunden
5010 :
5020 WINDOW 20,39,12,19
5022 PAPER 1:PEN 0:CLS
5028 PRINT
5030 PRINT " Ausgang gefunden"
5035 PRINT
5040 PRINT s;" Durchgaenge."
5100 PRINT
5110 PRINT" Druecken Sie eine   Taste..."
5120 k$=INKEY$:IF k$="" THEN 5120
5200 RETURN
5500 :
6000 REM Ausdruck
6002 LOCATE 1,1
6005 FOR y=1 TO 16
6010   FOR x=1 TO 15
6020     PRINT f$(y,x);
6030   NEXT x
6040  PRINT
6050 NEXT y
6070 x=2:y=2
6080 REM Startposition der Maus
6090 mar=0
6100 RETURN
```

4.3 Der MMC-RND-Lösungsalgorithmus

Der in diesem Robotics-Simulationsprogramm verwendete Algorithmus wendet im Vergleich zum MMC-BRAIN-Programm eine völlig andere Lösungsstrategie an: zunächst wird eine Marschrichtung gewählt. Sie wird so lange befolgt, bis die Maus auf eine Wegverzweigung trifft. Hier wählt sie einen der möglichen Wege, vermeidet es aber, den alten Weg, auf dem sie gekommen ist, zurückzuwandern.
Stellt die Micro-Mouse fest, daß sie sich in einer Sackgasse befindet, so geht sie ihren Weg bis zur letzten Verzweigung zurück. Dort angekommen, wählt sie eine neue Wegrichtung.
Die Schritte werden so lange wiederholt, bis der Ausgang des Labyrinths gefunden wird.
Schauen wir uns die Problemlösung im Programm an:

Der Supervisor

Er besteht wie der MMC-BRAIN-Programmkopf aus drei Prozeduren:

```
1 REM Kuenstliche Intelligenz auf dem CPC
3 REM Titel: MMC-RND
4 REM Micromouse Contest
5 :
6   MODE 1
7   PAPER 0
8   PEN 1:CLS
9   BORDER 9,9
10  WIDTH 60
20 :
90 REM ************ Supervisor **************
100     GOSUB 1000
110         REM Initialisation
200     GOSUB 2000
210         REM Hauptprogramm
300     GOSUB 5000
310         REM Ausgang gefunden
840     CLS
960 STOP
```

Die Initialisierung

Hier werden ein neues Labyrinth in das Feld eingelesen und eine neue Startposition für die Maus festgelegt:

```
1000 REM ********* Initialisation ***********
1005 :
1010 CLS
1020     PRINT TAB(21) "MMC-RND"
1030     PRINT
1040     PRINT TAB(21) "(c) Olaf Hartwig"
1045     PRINT: PRINT TAB(21) "1985"
1050 :
1060       ORIGIN 300,300:DRAW 330,1
```

```
1070        ORIGIN 300,130:DRAW 330,1
1075        ORIGIN 300,1:DRAW 1,400
1080        ORIGIN 1,90:DRAW 300,1
1085 :
1090  REM -------- Labyrinth lesen ----------
1100   DIM f$(17,15)
1110     FOR y=1 TO 16
1120       READ hi$
1130         FOR x=1 TO 15
1140           f$(y,x)=MID$(hi$,x,1)
1150         NEXT x
1160     NEXT y
1200  s=0
1210  REM Schleifenzaehler
1220 :
1280 RETURN
1400 :
1500 REM Labyrinth
1600       DATA ***************
1610       DATA * *           *
1620       DATA *   * **** * **
1630       DATA * **   *   *  *
1640       DATA *      *   *  *
1650       DATA * *           *
1660       DATA * ******  ** **
1670       DATA *      *      *
1680       DATA *****  **** ***
1690       DATA *      *      *
1700       DATA ***  ****  ** *
1710       DATA *             *
1720       DATA * ***  *** * **
1730       DATA * ***  *      *
1740       DATA *      *   *  *
1750       DATA **   **********
```

Das Hauptprogramm

Das Hauptprogramm beinhaltet den eigentlichen Lösungsalgorithmus.
Zunächst wird die Grundrichtung Süden vorgewählt. In den Programmzeilen 2120 bis 2150 wird abhängig vom Inhalt der Richtungsvariablen »ri$« eine Unterroutine der entsprechenden Richtung gewählt. Ist die Y-Koordinate der Mausposition gleich 16, so ist der Ausgang gefunden und das Programm springt zurück in den Programmkopf.
Der Programmteil lautet:

```
2000 REM ********* Hauptprogramm ************
2002 GOSUB 6000
2005 :
2010 ri$="s"
2020 REM Anfangsrichtung Sueden
2030 :
2100 REM ----- Richtungsroutinen waehlen ----
2105 ya=y:xa=x
2110 :
```

```
2115 m=0
2118 REM Marker fuer Sackgasse reset
2120    IF ri$="n" THEN GOSUB 2300
2130    IF ri$="w" THEN GOSUB 2400
2140    IF ri$="s" THEN GOSUB 2500
2150    IF ri$="o" THEN GOSUB 2600
2160 :
2170 IF m=9 THEN ri$=rk$:rh$=ri$
2200 IF y=16 THEN RETURN
2210 REM     Ausgang gefunden
2220 ri$=rh$
2250 GOTO 4000
```

Das Unterprogramm »Norden« wird bei der entsprechend eingeschlagenen Richtung der Maus aufgerufen. Zunächst wird überprüft, ob sich die Maus in einer Sackgasse befindet. Ist das der Fall, so wird die Richtung geändert und aus der Routine zurückgesprungen.
Anschließend erfolgt die Erfassung der Umgebung. Dabei wird untersucht, ob die gewählte Richtung durch eine Wand behindert wird.
Das Programm-Modul sieht wie folgt aus:

```
2300 REM --------- Richtung Norden ----------
2310     IF f$(y-1,x)="*" AND f$(y,x-1)="*" AND f$(y,x+1)="*"
         THEN rk$="s":m=9:RETURN
2315     REM Befindet sich die Maus in einer Sackgasse?
2317 :
2320     GOSUB 2700
2330        IF rh$="s" THEN rh$="n"
2335        REM Alte Richtung zurueck
2337 :
2340          IF f$(y,x-1)="*" AND rh$="w" THEN 2320
2345          IF f$(y,x+1)="*" AND rh$="o" THEN 2320
2350          IF f$(y-1,x)="*" AND rh$="n" THEN 2320
2355 :
2360     GOSUB 2800
2370     REM Wegrichtung bestimmen
2380 RETURN
```

Die Module für die Richtungen »Westen«, »Süden« und »Osten« sind entsprechend aufgebaut. Sie lauten wie folgt:

```
2400 REM --------- Richtung Westen ----------
2410     IF f$(y-1,x)="*" AND f$(y,x-1)="*" AND f$(y+1,x)="*"
         THEN rk$="o":m=9:RETURN
2415     REM Pruefung auf Sackgasse
2417 :
2420     GOSUB 2700
2437 :
2440          IF f$(y-1,x)="*" AND rh$="n" THEN 2420
2445          IF f$(y,x-1)="*" AND rh$="w" THEN 2420
2450          IF f$(y+1,x)="*" AND rh$="s" THEN 2420
2455 :
2460     GOSUB 2800
```

```
2470     REM Wegrichtung bestimmen
2480  RETURN
2495  :
2500  REM --------- Richtung Sueden ----------
2510     IF f$(y+1,x)="*" AND f$(y,x-1)="*" AND f$(y,x+1)="*"
         THEN rk$="n":m=9:RETURN
2515     REM Befindet sich die Maus in einer Sackgasse?
2517  :
2520     GOSUB 2700
2530        IF rh$="n" THEN rh$="s"
2535        REM Alte Richtung zurueck
2537  :
2540          IF f$(y,x-1)="*" AND rh$="w" THEN 2520
2545          IF f$(y,x+1)="*" AND rh$="o" THEN 2520
2550          IF f$(y+1,x)="*" AND rh$="s" THEN 2520
2555  :
2560     GOSUB 2800
2570     REM Wegrichtung bestimmen
2580  RETURN
2590  :
2600  REM --------- Richtung Osten -----------
2610     IF f$(y-1,x)="*" AND f$(y+1,x)="*" AND f$(y,x+1)="*"
         THEN rk$="w":m=9:RETURN
2615     REM befindet sich die maus in einer sackgasse?
2617  :
2620     GOSUB 2700
2637  :
2640          IF f$(y+1,x)="*" AND rh$="s" THEN 2620
2645          IF f$(y-1,x)="*" AND rh$="n" THEN 2620
2650          IF f$(y,x+1)="*" AND rh$="o" THEN 2620
2655  :
2660     GOSUB 2800
2670     REM Wegrichtung bestimmen
2680  RETURN
2690  :
```

Das Programm verwendet die folgenden Unterroutinen, die dazu dienen, die Richtungen intern zu repräsentieren und umzuwandeln:

```
2700  REM -------- Richtung bestimmen --------
2710    ri=1+INT(RND(1)*4)
2730       IF ri=1 THEN rh$="n"
2740       IF ri=2 THEN rh$="w"
2750       IF ri=3 THEN rh$="s"
2760       IF ri=4 THEN rh$="o"
2790  RETURN
2795  :
2800  REM --- Wegrichtung in rh$ festlegen ---
2820       IF rh$="n" AND y>1  THEN y=y-1
2830       IF rh$="s" AND y<16 THEN y=y+1
2840       IF rh$="w" AND x>1  THEN x=x-1
2850       IF rh$="o" AND x<14 THEN x=x+1
2870  RETURN
```

Wie bei MMC-BRAIN wird dann das Labyrinth ausgegeben und die Position der Maus angezeigt:

```
4000 REM *** Ausgabe der Labyrinthposition **
4005  s=s+1
4007  REM Durchgaenge zaehlen
4010   zaehler=0
4050   LOCATE xa,ya: PRINT " ";
4152 :
4153     LOCATE 21,17
4155     LOCATE 21,19: PRINT "Richtung___> ";rh$
4157     LOCATE 21,21
4158 :
4159       IF rh$="s" THEN PRINT "   Sueden"
4161       IF rh$="o" THEN PRINT "   Osten "
4163       IF rh$="n" THEN PRINT "   Norden"
4165       IF rh$="w" THEN PRINT "   Westen"
4170 :
4175     LOCATE 21,23: PRINT "Durchgang Nr.";s
4180 :
4200   REM Mausposition
4220   LOCATE x,y
4230   PRINT "M"
4240 :
4250  LOCATE 1,1
4270 :
4400 GOTO 2100
```

Beim ersten Durchgang wird in der Zeile 2002 das Unterprogramm aufgerufen, das das gesamte Labyrinth ausgibt und die Startposition der Maus festlegt.
Es sieht wie folgt aus:

```
6000 REM ******* Labyrinth ausdrucken *******
6050 LOCATE 1,1
6100  FOR i=1 TO 16
6110      FOR k=1 TO 15
6120          PRINT f$(i,k);
6130      NEXT k
6140    PRINT
6150  NEXT i
6200   y=6: x=2
6205   xa=x: ya=x
6210   REM Startposition der Maus
6300 RETURN
```

Die Endprozedur

Sobald der Ausgang des Labyrinths von der Maus gefunden wurde, wird ein Window eröffnet, in dem eine entsprechende Meldung ausgegeben wird.
Die Prozedur lautet:

```
5000 REM ******* gefundener Ausgang *********
5010 :
5020  WINDOW 1,18,21,25
5025   PAPER 1: PEN 0: CLS
5030     PRINT " Ausgang gefunden"
5040     PRINT s;" Durchgaenge."
5050   PRINT" Druecken Sie eine Taste"
5060   q$=INKEY$: IF q$="" THEN 5060
5100  PRINT
5200 RETURN
```

Wie Sie bei einem Programmlauf schnell feststellen werden, ist dieser Algorithmus effektiver als MMC-BRAIN, obwohl hier kein »Gedächtnis« über bereits begangene Wege vorhanden ist. Es ist auch nicht erforderlich, da ein »Hängenbleiben« in einer Endlosschleife hier nicht möglich ist.

Das komplette MMC-RND-Programm im vollständigen Zusammenhang und in der korrekten Reihenfolge der einzelnen Module lautet wie folgt:

```
1 REM Kuenstliche Intelligenz auf dem CPC
3 REM Titel: MMC-RND
4 REM Micromouse Contest
5 :
6   MODE 1
7   PAPER 0
8   PEN 1:CLS
9   BORDER 9,9
10  WIDTH 60
20 :
90 REM ************ Supervisor **************
100      GOSUB 1000
110          REM Initialisation
200      GOSUB 2000
210          REM Hauptprogramm
300      GOSUB 5000
310          REM Ausgang gefunden
840      CLS
960 STOP
970 :
980 REM Prozeduren
990 :
1000 REM ********* Initialisation ***********
1005 :
1010 CLS
1020     PRINT TAB(21) "MMC-RND"
1030     PRINT
1040     PRINT TAB(21) "(c) Olaf Hartwig"
1045     PRINT: PRINT TAB(21) "1985"
1050 :
1060       ORIGIN 300,300:DRAW 330,1
1070       ORIGIN 300,130:DRAW 330,1
1075       ORIGIN 300,1:DRAW 1,400
1080       ORIGIN 1,90:DRAW 300,1
1085 :
1090  REM -------- Labyrinth lesen ----------
```

```
1100   DIM f$(17,15)
1110     FOR y=1 TO 16
1120       READ hi$
1130         FOR x=1 TO 15
1140           f$(y,x)=MID$(hi$,x,1)
1150         NEXT x
1160     NEXT y
1200  s=0
1210  REM Schleifenzaehler
1220 :
1280 RETURN
1400 :
1500 REM Labyrinth
1600      DATA ***************
1610      DATA * *           *
1620      DATA *   * **** * **
1630      DATA * **   *   *  *
1640      DATA *      *   *  *
1650      DATA * *           *
1660      DATA * ******  ** **
1670      DATA *      *      *
1680      DATA *****  **** ***
1690      DATA *      *      *
1700      DATA ***  ****  ** *
1710      DATA *             *
1720      DATA * ***  *** * **
1730      DATA * ***  *      *
1740      DATA *      *   *  *
1750      DATA **   **********
1780 :
2000 REM ********* Hauptprogramm ************
2002 GOSUB 6000
2005 :
2010 ri$="s"
2020 REM Anfangsrichtung Sueden
2030 :
2100 REM ----- Richtungsroutinen waehlen ----
2105 ya=y:xa=x
2110 :
2115 m=0
2118 REM Marker fuer Sackgasse reset
2120   IF ri$="n" THEN GOSUB 2300
2130   IF ri$="w" THEN GOSUB 2400
2140   IF ri$="s" THEN GOSUB 2500
2150   IF ri$="o" THEN GOSUB 2600
2160 :
2170 IF m=9 THEN ri$=rk$:rh$=ri$
2200 IF y=16 THEN RETURN
2210 REM     Ausgang gefunden
2220 ri$=rh$
2250 GOTO 4000
2270 :
2300 REM --------- Richtung Norden ----------
2310    IF f$(y-1,x)="*" AND f$(y,x-1)="*" AND f$(y,x+1)="*"
        THEN rk$="s":m=9:RETURN
2315    REM Befindet sich die Maus in einer Sackgasse?
```

```
2317 :
2320     GOSUB 2700
2330        IF rh$="s" THEN rh$="n"
2335        REM Alte Richtung zurueck
2337 :
2340          IF f$(y,x-1)="*" AND rh$="w" THEN 2320
2345          IF f$(y,x+1)="*" AND rh$="o" THEN 2320
2350          IF f$(y-1,x)="*" AND rh$="n" THEN 2320
2355 :
2360     GOSUB 2800
2370     REM Wegrichtung bestimmen
2380 RETURN
2395 :
2400 REM --------- Richtung Westen ----------
2410     IF f$(y-1,x)="*" AND f$(y,x-1)="*" AND f$(y+1,x)="*"
         THEN rk$="o":m=9:RETURN
2415     REM Pruefung auf Sackgasse
2417 :
2420     GOSUB 2700
2437 :
2440          IF f$(y-1,x)="*" AND rh$="n" THEN 2420
2445          IF f$(y,x-1)="*" AND rh$="w" THEN 2420
2450          IF f$(y+1,x)="*" AND rh$="s" THEN 2420
2455 :
2460     GOSUB 2800
2470     REM Wegrichtung bestimmen
2480 RETURN
2495 :
2500 REM --------- Richtung Sueden ----------
2510     IF f$(y+1,x)="*" AND f$(y,x-1)="*" AND f$(y,x+1)="*"
         THEN rk$="n":m=9:RETURN
2515     REM Befindet sich die Maus in einer Sackgasse?
2517 :
2520     GOSUB 2700
2530        IF rh$="n" THEN rh$="s"
2535        REM Alte Richtung zurueck
2537 :
2540          IF f$(y,x-1)="*" AND rh$="w" THEN 2520
2545          IF f$(y,x+1)="*" AND rh$="o" THEN 2520
2550          IF f$(y+1,x)="*" AND rh$="s" THEN 2520
2555 :
2560     GOSUB 2800
2570     REM Wegrichtung bestimmen
2580 RETURN
2590 :
2600 REM --------- Richtung Osten -----------
2610     IF f$(y-1,x)="*" AND f$(y+1,x)="*" AND f$(y,x+1)="*"
         THEN rk$="w":m=9:RETURN
2615     REM befindet sich die maus in einer sackgasse?
2617 :
2620     GOSUB 2700
2637 :
2640          IF f$(y+1,x)="*" AND rh$="s" THEN 2620
2645          IF f$(y-1,x)="*" AND rh$="n" THEN 2620
2650          IF f$(y,x+1)="*" AND rh$="o" THEN 2620
2655 :
```

```
2660    GOSUB 2800
2670    REM Wegrichtung bestimmen
2680 RETURN
2690 :
2700 REM -------- Richtung bestimmen --------
2710   ri=1+INT(RND(1)*4)
2730      IF ri=1 THEN rh$="n"
2740      IF ri=2 THEN rh$="w"
2750      IF ri=3 THEN rh$="s"
2760      IF ri=4 THEN rh$="o"
2790 RETURN
2795 :
2800 REM --- Wegrichtung in rh$ festlegen ---
2820      IF rh$="n" AND y>1  THEN y=y-1
2830      IF rh$="s" AND y<16 THEN y=y+1
2840      IF rh$="w" AND x>1  THEN x=x-1
2850      IF rh$="o" AND x<14 THEN x=x+1
2870 RETURN
2880 :
2890 :
4000 REM *** Ausgabe der Labyrinthposition **
4005  s=s+1
4007  REM Durchgaenge zaehlen
4010   zaehler=0
4050   LOCATE xa,ya: PRINT " ";
4152 :
4153     LOCATE 21,17
4155     LOCATE 21,19: PRINT "Richtung___> ";rh$
4157     LOCATE 21,21
4158 :
4159       IF rh$="s" THEN PRINT "   Sueden"
4161       IF rh$="o" THEN PRINT "   Osten "
4163       IF rh$="n" THEN PRINT "   Norden"
4165       IF rh$="w" THEN PRINT "   Westen"
4170 :
4175     LOCATE 21,23: PRINT "Durchgang Nr.";s
4180 :
4200   REM Mausposition
4220   LOCATE x,y
4230   PRINT "M"
4240 :
4250  LOCATE 1,1
4270 :
4400 GOTO 2100
4440 :
5000 REM ******* gefundener Ausgang *********
5010 :
5020  WINDOW 1,18,21,25
5025   PAPER 1: PEN 0: CLS
5030     PRINT " Ausgang gefunden"
5040     PRINT s;" Durchgaenge."
5050   PRINT" Druecken Sie eine Taste"
5060   q$=INKEY$: IF q$="" THEN 5060
5100  PRINT
5200 RETURN
5500 :
```

```
6000 REM ******* Labyrinth ausdrucken *******
6050 LOCATE 1,1
6100  FOR i=1 TO 16
6110      FOR k=1 TO 15
6120          PRINT f$(i,k);
6130      NEXT k
6140    PRINT
6150  NEXT i
6200   y=6: x=2
6205   xa=x: ya=x
6210   REM Startposition der Maus
6300 RETURN
```

4.4 Der PLEDGE-Algorithmus

Dieser Algorithmus wurde vom zwölfjährigen John Pledge aus England entwickelt. Er stellt eine gute Lösungsstrategie dar. Mit ihm findet nahezu jede Maus auf kürzestem Weg aus einem Labyrinth.

Die Regeln des »Pledge-Algorithmus« lauten folgendermaßen:

1. Wähle eine Anfangsrichtung und richte dich nach dieser Richtung aus.
2. Gehe geradeaus, bis du auf ein Hindernis triffst.
3. Drehe dich dann so lange nach links, bis sich das Hindernis rechts von dir befindet.
4. Bewege dich um das Hindernis herum, so daß es stets rechts von dir bleibt, bis die Gesamtdrehung gleich Null ist.
5. Gehe zurück zu Schritt 2.

Die Anwendung dieser Regeln in einem Beispiel-Labyrinth ist in Bild 4.1 dargestellt:

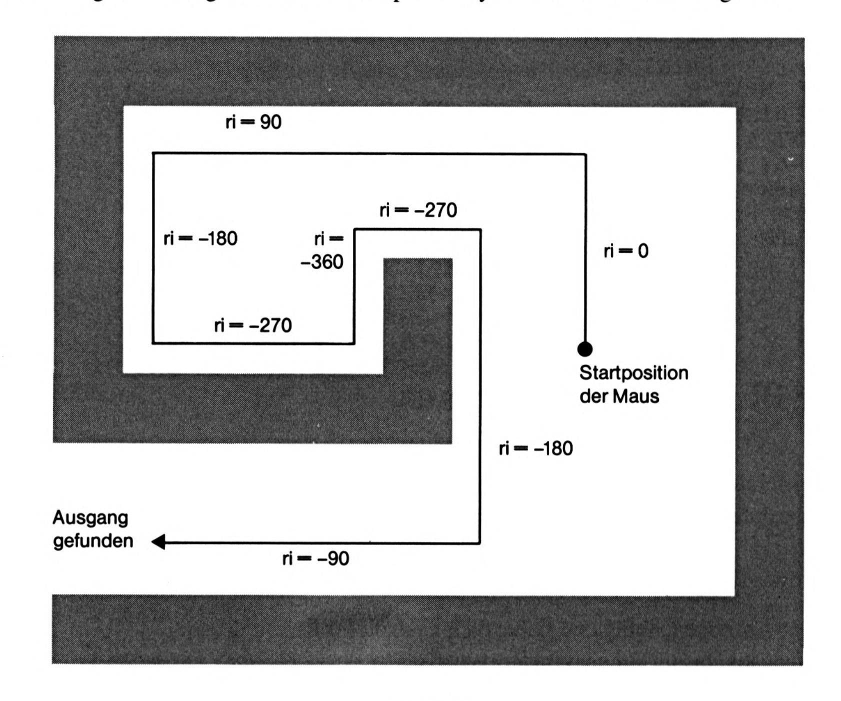


Bild 4.1

Der Algorithmus wurde folgendermaßen auf dem Computer realisiert:

Der Supervisor

Er besteht aus den drei Prozeduren »Initialisierung«, »Hauptprogramm« und »Ausgang«. Dabei befindet sich der entscheidende Programmteil in dem Modul »Hauptprogramm«. Die einzelnen Module werden nacheinander aufgerufen:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: Pledge-Algorithmus
5 REM Micromouse Contest
6 :
10     MODE 1
15     PAPER 1: PEN 0
20     CLS
25     BORDER 9,9
30     WIDTH 60
40 :
90 REM ************ Supervisor *************
100        GOSUB 1000
110            REM Initialisation
200        GOSUB 2000
```

```
210          REM Hauptprogramm
300       GOSUB 5000
310          REM Ausgang gefunden
840       CLS
950 :
960 STOP
```

Die Initialisierung

Nachdem der Programmtitel auf dem Monitor angezeigt und zwei Trennungslinien gezeichnet wurden, liest das Programm den Irrgarten aus DATAs in das Feld »f$(x,y)« ein. Anschließend werden die Startkoordinaten der Maus bestimmt und der Schleifenzähler auf Null gesetzt:

```
1000 REM ********* Initialisation **********
1005 :
1010 CLS
1020   PRINT TAB(20) "MAUSLABYRINTH PLEDGE"
1030   PRINT
1040   PRINT TAB(20) "(c) Olaf Hartwig"
1041   PRINT: PRINT TAB(20) "1986"
1042      ORIGIN 280,1: DRAW 0,400,0
1044      ORIGIN 280,300: DRAW 360,0,0
1050 :
1090  REM -------- Labyrinth einlesen ------
1100   DIM f$(17,15)
1110     FOR y=1 TO 16
1120       READ hi$
1130         FOR x=1 TO 15
1140           f$(y,x)=MID$(hi$,x,1)
1150         NEXT x
1160     NEXT y
1200  s=0
1210  REM Schleifenzaehler
1220 :
1230 x=2
1240 y=6
1250 ax=x
1255 ay=y
1260 REM Startposition der Maus
1270 :
1280 RETURN
1300 :
1500 REM ------------ Labyrinth ------------
1600      DATA ***************
1610      DATA * *   *    *  *
1620      DATA *   * **** * **
1630      DATA * **   *   *  *
1640      DATA * *    *   *  *
1650      DATA * *           *
1660      DATA * ***** **** **
1670      DATA *       *     *
1680      DATA *****  **** ***
1690      DATA *             *
1700      DATA *** *** ***** *
```

```
1710       DATA *     *       *
1720       DATA * ***   ** * **
1730       DATA * *      * *  *
1740       DATA *      *   *  *
1750       DATA **  ***********
```

Der Haupt-Lösungsalgorithmus

Zunächst wird die Anfangsrichtung der Maus mit ri=0 vorgewählt. Als Richtungswerte kommen

0 bzw. –360	für Norden,
–90	für Westen,
–180	für Süden sowie
–270	für Osten

in Frage. Dann wird geprüft, ob ein Hindernis, also eine Wand, im Weg ist. Ist das nicht der Fall, so erfolgt ein weiterer Schritt nach Norden. Befindet sich jedoch ein Hindernis vor der Maus, so muß sie sich so lange nach links drehen, bis sich das Hindernis rechts von ihr befindet.
Dieser Teilalgorithmus befindet sich in den Zeilen 2000 bis 2830. Zunächst wird der Kurs der Maus nach Westen korrigiert. Je nach aktueller Marschrichtung wird eine von vier Richtungs-Unterroutinen aufgerufen.
Das Programm-Modul lautet:

```
2000 REM ********** Hauptprogramm **********
2005    WINDOW 1,17,1,24
2007    PEN 1: PAPER 0: CLS
2010 CLS
2015 GOSUB 4000
2020 :
2040 ri=0
2050 REM Anfangsrichtung vorwaehlen
2090 :
2100 IF f$(y-1,x)=" " THEN y=y-1: GOSUB 4152:
     GOTO 2100: REM kein Hindernis
2150 :
2200 ri=ri-90
2210 REM Drehe dich nach links, bis sich das Hindernis
         rechts befindet
2250 :
2300 IF y>=15 THEN RETURN: REM Ausgang gefunden
```

Die folgenden vier Richtungsmodule testen, ob sich ein Hindernis in Marschrichtung befindet. Wenn ja, so wird die Drehung der Maus fortgesetzt.
Befindet sich keine Wand in der gewählten Richtung, testet das Programm, ob das Hindernis rechts von der Maus liegt. Ist das der Fall, dann bewegt sich die Maus in dieser Richtung weiter durch das Labyrinth.
Auf diese Weise wird ein optimaler Weg aus dem Labyrinth gefunden.

```
2400 REM Schleife
2410 IF ri=-90 THEN GOTO 2500
2420 IF ri=-180 THEN GOTO 2600
2430 IF ri=-270 THEN GOTO 2700
2440 IF ri=-360 THEN GOTO 2800
2450 IF ri=0 THEN 2100
2460 GOTO 2300
2470 REM Bewege dich um das Hindernis herum, so dass es stets
         rechts von dir bleibt
2490 :
2500 REM ri=-90
2510 IF f$(y,x-1)=" " AND f$(y-1,x)<>" " THEN x=x-1:
     GOSUB 4152:GOTO 2400
2520 IF f$(y-1,x)=" " THEN y=y-1: ri=ri+90: GOSUB 4152: GOTO 2400
2530 IF f$(y,x-1)="*" THEN ri=ri-90: GOTO 2400
2590 '
2600 REM ri=-180
2610 IF f$(y+1,x)=" " AND f$(y,x-1)<>" " THEN y=y+1:
     GOSUB 4152:GOTO 2400
2620 IF f$(y.,x-1)=" " THEN x=x-1: ri=ri+90: GOSUB 4152: GOTO 2400
2630 IF f$(y+1,x)="*" THEN ri=ri-90: GOTO 2400
2690 '
2700 REM ri=-270
2710 IF f$(y,x+1)=" " AND f$(y+1,x)<>" " THEN x=x+1:
     GOSUB 4152:GOTO 2400
2720 IF f$(y+1,x)=" " THEN y=y+1: ri=ri+90: GOSUB 4152: GOTO 2400
2730 IF f$(y,x+1)="*" THEN ri=ri-90: GOTO 2400
2790 '
2800 REM ri=-360
2810 IF f$(y-1,x)=" " AND f$(y,x+1)<>" " THEN y=y-1:
     GOSUB 4152:GOTO 2400
2820 IF f$(y,x+1)=" " THEN x=x+1: ri=ri+90: GOSUB 4152: GOTO 2400
2830 IF f$(y,x-1)="*" THEN PRINT "Kein Ausgang!": STOP
2890 '
```

Anschließend wird der Irrgarten zusammen mit der aktuellen Position der Maus sowie der gerade eingeschlagenen Richtung und der aktuellen Durchgangsnummer auf dem Bildschirm angezeigt:

```
4000 REM ****** Ausdruck des Labyrinths ****
4050 LOCATE 1,1
4100  FOR i=1 TO 16
4110    FOR k=1 TO 15
4120       PRINT f$(i,k);
4130    NEXT k
4140   PRINT
4150  NEXT i
4151 RETURN
4152 REM Statusangaben
4153 LOCATE 1,19
4155   PRINT "Richtung_> ";ri
4157   PRINT
4158 :
4159      IF ri=-180 THEN PRINT "   Sueden"
4161      IF ri=-270 THEN PRINT "   Osten "
```

```
4163       IF ri=   0 THEN PRINT "    Norden"
4165       IF ri= -90 THEN PRINT "    Westen"
4166 :
4167 durchgang=durchgang+1
4168 :
4170  PRINT
4175  PRINT "Durchgang ";durchgang
4180 :
4200 REM Mausposition
4205 LOCATE ax,ay
4210 PRINT " "
4220  LOCATE x,y
4230  PRINT"M"
4240 :
4300 ax=x
4310 ay=y
4400 RETURN
```

Das Programmende

Sobald der Ausgang des Labyrinths gefunden wurde, wird ein Fenster definiert und eine kurze Meldung sowie die Anzahl der benötigten Durchgänge angezeigt:

```
5000 REM ********* Ausgang gefunden ********
5010 :
5020  WINDOW 16,40,14,22
5025   PEN 0: PAPER 2: CLS
5030     PRINT:PRINT " Ausgang gefunden"
5040     PRINT:PRINT durchgang;" Durchgaenge."
5100 :
5110     PRINT:PRINT:PRINT " Druecken Sie eine Taste"
5120     w$=INKEY$:IF w$="" THEN 5120
5150 :
5200 RETURN
```

Das komplette Programm

An dieser Stelle erhalten Sie die komplette und zusammenhängende Version des Pledge-Algorithmus:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: Pledge-Algorithmus
5 REM Micromouse Contest
6 :
10    MODE 1
15    PAPER 1: PEN 0
20    CLS
25    BORDER 9,9
30    WIDTH 60
40 :
90 REM ************ Supervisor *************
100       GOSUB 1000
110           REM Initialisation
```

```
200        GOSUB 2000
210            REM Hauptprogramm
300        GOSUB 5000
310            REM Ausgang gefunden
840        CLS
950 :
960 STOP
970 :
980 REM Prozeduren
990 :
1000 REM ********* Initialisation **********
1005 :
1010 CLS
1020   PRINT TAB(20) "MAUSLABYRINTH PLEDGE"
1030   PRINT
1040   PRINT TAB(20) "(c) Olaf Hartwig"
1041   PRINT: PRINT TAB(20) "1986"
1042      ORIGIN 280,1: DRAW 0,400,0
1044      ORIGIN 280,300: DRAW 360,0,0
1050 :
1090  REM -------- Labyrinth einlesen ------
1100  DIM f$(17,15)
1110    FOR y=1 TO 16
1120      READ hi$
1130        FOR x=1 TO 15
1140          f$(y,x)=MID$(hi$,x,1)
1150        NEXT x
1160    NEXT y
1200  s=0
1210  REM Schleifenzaehler
1220 :
1230 x=2
1240 y=6
1250 ax=x
1255 ay=y
1260 REM Startposition der Maus
1270 :
1280 RETURN
1300 :
1500 REM ------------ Labyrinth ------------
1600      DATA ***************
1610      DATA * *   *    *  *
1620      DATA *   * **** * **
1630      DATA * **   *   *  *
1640      DATA * *    *   *  *
1650      DATA * *           *
1660      DATA * ***** **** **
1670      DATA *       *     *
1680      DATA *****  **** ***
1690      DATA *             *
1700      DATA *** *** ***** *
1710      DATA *     *       *
1720      DATA * ***   ** * **
1730      DATA * *      * *  *
1740      DATA *      *   *  *
```

```
1750      DATA **   ***********
1800 :
2000 REM ********** Hauptprogramm **********
2005    WINDOW 1,17,1,24
2007    PEN 1: PAPER 0: CLS
2010 CLS
2015 GOSUB 4000
2020 :
2040 ri=0
2050 REM Anfangsrichtung vorwaehlen
2090 :
2100 IF f$(y-1,x)=" " THEN y=y-1: GOSUB 4152:
     GOTO 2100: REM kein Hindernis
2150 :
2200 ri=ri-90
2210 REM Drehe dich nach links, bis sich das Hindernis
          rechts befindet
2250 :
2300 IF y>=15 THEN RETURN: REM Ausgang gefunden
2350 :
2400 REM Schleife
2410 IF ri=-90 THEN GOTO 2500
2420 IF ri=-180 THEN GOTO 2600
2430 IF ri=-270 THEN GOTO 2700
2440 IF ri=-360 THEN GOTO 2800
2450 IF ri=0 THEN 2100
2460 GOTO 2300
2470 REM Bewege dich um das Hindernis herum, so dass es stets
         rechts von dir bleibt
2490 :
2500 REM ri=-90
2510 IF f$(y,x-1)=" " AND f$(y-1,x)<>" " THEN x=x-1:
     GOSUB 4152:GOTO 2400
2520 IF f$(y-1,x)=" " THEN y=y-1: ri=ri+90: GOSUB 4152: GOTO 2400
2530 IF f$(y,x-1)="*" THEN ri=ri-90: GOTO 2400
2590 '
2600 REM ri=-180
2610 IF f$(y+1,x)=" " AND f$(y,x-1)<>" " THEN y=y+1:
     GOSUB 4152:GOTO 2400
2620 IF f$(y,x-1)=" " THEN x=x-1: ri=ri+90: GOSUB 4152: GOTO 2400
2630 IF f$(y+1,x)="*" THEN ri=ri-90: GOTO 2400
2690 '
2700 REM ri=-270
2710 IF f$(y,x+1)=" " AND f$(y+1,x)<>" " THEN x=x+1:
     GOSUB 4152:GOTO 2400
2720 IF f$(y+1,x)=" " THEN y=y+1: ri=ri+90: GOSUB 4152: GOTO 2400
2730 IF f$(y,x+1)="*" THEN ri=ri-90: GOTO 2400
2790 '
2800 REM ri=-360
2810 IF f$(y-1,x)=" " AND f$(y,x+1)<>" " THEN y=y-1:
     GOSUB 4152:GOTO 2400
2820 IF f$(y,x+1)=" " THEN x=x+1: ri=ri+90: GOSUB 4152: GOTO 2400
2830 IF f$(y,x-1)="*" THEN PRINT "Kein Ausgang!": STOP
2890 '
4000 REM ****** Ausdruck des Labyrinths ****
4050 LOCATE 1,1
```

```
4100  FOR i=1 TO 16
4110    FOR k=1 TO 15
4120       PRINT f$(i,k);
4130    NEXT k
4140   PRINT
4150  NEXT i
4151 RETURN
4152 REM Statusangaben
4153 LOCATE 1,19
4155   PRINT "Richtung_> ";ri
4157   PRINT
4158 :
4159      IF ri=-180 THEN PRINT "   Sueden"
4161      IF ri=-270 THEN PRINT "   Osten "
4163      IF ri=   0 THEN PRINT "   Norden"
4165      IF ri= -90 THEN PRINT "   Westen"
4166 :
4167 durchgang=durchgang+1
4168 :
4170  PRINT
4175  PRINT "Durchgang ";durchgang
4180 :
4200 REM Mausposition
4205 LOCATE ax,ay
4210 PRINT " "
4220  LOCATE x,y
4230  PRINT"M"
4240 :
4300 ax=x
4310 ay=y
4400 RETURN
4440 :
5000 REM ********* Ausgang gefunden ********
5010 :
5020  WINDOW 16,40,14,22
5025   PEN 0: PAPER 2: CLS
5030     PRINT:PRINT " Ausgang gefunden"
5040     PRINT:PRINT durchgang;" Durchgaenge."
5100 :
5110     PRINT:PRINT:PRINT " Druecken Sie eine Taste"
5120     w$=INKEY$:IF w$="" THEN 5120
5150 :
5200 RETURN
```

Der Pledge-Algorithmus ist, wie Sie an dem Programm gesehen haben, sehr effektiv. Testen Sie einmal die volle Leistungsfähigkeit der Lösungsstrategie anhand eines Programms mit einem umfangreicheren Labyrinth.

4.5 Das MAXI-PLEDGE-Programm

Im folgenden finden Sie ein Beispiel-Listing mit allen Änderungen für ein Labyrinth mit einer Größe von 39 mal 23 (=897) Feldern. Zum Vergleich: der alte Irrgarten umfaßte nur 255 Felder. Sie können anhand des Listings deutlich sehen, wie das Pledge-Programm für beliebige Irrgärten mit frei wählbaren Labyrinthgrößen umgewandelt werden kann.
Das vollständige Listing des MAXI-PLEDGE-Programms lautet folgendermaßen:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: MAXI-PLEDGE
5 REM Micromouse Contest
7 :
10   MODE 1
30   PAPER 0
40   PEN 1
50   BORDER 9
60 :
90 REM ************** Supervisor ***************
100        GOSUB 1000
110            REM Initialisation
200        GOSUB 2000
210            REM Hauptprogramm
300        GOSUB 5000
310            REM Ausgang gefunden
840        CLS
950 :
960 STOP
970 :
980 REM Prozeduren
990 :
1000 REM *********** Initialisation *************
1005 :
1010 CLS
1050 :
1090  REM Labyrinth lesen
1100   DIM f$(25,39)
1110     FOR y=1 TO 23
1120       READ hi$
1130         FOR x=1 TO 39
1140           f$(y,x)=MID$(hi$,x,1)
1150         NEXT x
1160     NEXT y
1200  s=0
1220 :
1230 x=2
1240 y=6: x=30
1250 ax=x
1255 ay=y
1258 IF marker=0 THEN GOSUB 4000
1260 REM Startposition der Maus
1270 :
```

```
1280 RETURN
1300 :
1500 REM ------------- Labyrinth ----------------
1600 DATA ***************************************
1610 DATA * *   *    *        *      *  **  *   *
1620 DATA *   * **** * **  *  ****  *  **     ***
1630 DATA * **   *   *  *  *        *   *       *
1640 DATA * *    *   *     *        **   ********
1650 DATA * *           *     ****   *          *
1660 DATA * ***** **** **     *  *              *
1670 DATA *       *         ***  *   *    *******
1680 DATA *****  **** ***  *     * ***  *       *
1690 DATA *             *        *  *   *********
1700 DATA *** *** ***** *    *****  *           *
1710 DATA *     *       *  **       *           *
1720 DATA * ***   ** * **   *      * ***  **   **
1730 DATA * *      * *  *   **   **  *   ***   **
1740 DATA *      *   *  **    ***   *      ******
1750 DATA *                                     *
1760 DATA *     *   *                         ***
1770 DATA *    *          **********    ***     *
1780 DATA * ****          *        *    *       *
1790 DATA * *        ******     ****         ****
1800 DATA * ****    *                      **   *
1810 DATA *         *****                       *
1820 DATA ***     *******************************
1900 :
1950 :
2000 REM ************ Hauptprogramm *************
2005 :
2250 ri=0
2255 :
2260 IF f$(y-1,x)=" " THEN y=y-1: GOSUB 4152: GOTO 2260:
     REM kein Hindernis
2270 REM Hindernis im Norden getroffen
2290 :
2300 ri=ri-90
2310 REM Drehe dich nach links, bis sich das Hindernis
     rechts befindet
2400 REM Bewege dich um das Hindernis herum, so dass es
     stets rechts von dir bleibt
2410 IF y>=22 THEN RETURN: 'Ausgang gefunden
2450 IF ri=-90 THEN GOTO 2500
2460 IF ri=-180 THEN GOTO 2600
2470 IF ri=-270 THEN GOTO 2700
2480 IF ri=-360 THEN GOTO 2800
2485 IF ri=0 THEN 2260
2490 GOTO 2450
2495 '
2500 REM ri=-90
2510 IF f$(y,x-1)=" " AND f$(y-1,x)<>" " THEN x=x-1:
     GOSUB 4152:GOTO 2450
2520 IF f$(y-1,x)=" " THEN y=y-1: ri=ri+90:GOSUB 4152:
     GOTO 2450
2530 IF f$(y,x-1)="*" THEN ri=ri-90:GOTO 2450
2595 '
```

```
2600 REM ri=-180
2610 IF f$(y+1,x)=" " AND f$(y,x-1)<>" " THEN y=y+1:
     GOSUB 4152:GOTO 2450
2620 IF f$(y,x-1)=" " THEN x=x-1: ri=ri+90:GOSUB 4152:
     GOTO 2450
2630 IF f$(y+1,x)="*" THEN ri=ri-90:GOTO 2450
2695 '
2700 REM ri=-270
2710 IF f$(y,x+1)=" " AND f$(y+1,x)<>" " THEN x=x+1:
     GOSUB 4152:GOTO 2450
2720 IF f$(y+1,x)=" " THEN y=y+1: ri=ri+90:GOSUB 4152:
     GOTO 2450
2730 IF f$(y,x+1)="*" THEN ri=ri-90:GOTO 2450
2795 '
2800 REM ri=-360
2810 IF f$(y-1,x)=" " AND f$(y,x+1)<>" " THEN y=y-1:
     GOSUB 4152:GOTO 2450
2820 IF f$(y,x+1)=" " THEN x=x+1: ri=ri+90:GOSUB 4152:
     GOTO 2450
2830 IF f$(y-1,x)="*" THEN PRINT "Kein Ausgang!": STOP
2895 '
3990 :
4000 REM ******** Ausdruck des Labyrinths *******
4010   marker=9
4050    LOCATE 1,1
4100      FOR i=1 TO 23
4110         FOR k=1 TO 39
4120            PRINT f$(i,k);
4130         NEXT k
4140       PRINT
4150      NEXT i
4151 RETURN
4152 REM Statusausgaben
4153 durchgang=durchgang+1
4154 LOCATE 1,24
4155 PRINT "Richtung___> ";ri;" ";
4159 '      IF ri=-180 THEN PRINT "   Sueden   ";
4161 '      IF ri=-270 THEN PRINT "   Osten    ";
4163 '      IF ri=0    THEN PRINT "   Norden   ";
4165 '      IF ri=-90  THEN PRINT "   Westen   ";
4169 PRINT
4170 :
4175 PRINT "Durchgang Nr.";durchgang
4180 :
4200 REM Mausposition
4205    LOCATE ax,ay
4210    PRINT " "
4220    LOCATE x,y
4230    PRINT "M"
4240 :
4300  ax=x
4310  ay=y
4350 RETURN
4400 STOP
4440 :
```

```
4450 :
5000 REM *********** Ausgang gefunden ***********
5010 :
5020  WINDOW 22,35,1,20
5022   PAPER 2
5024   PEN 3
5026    CLS
5040     PRINT: PRINT " MAXI-PLEDGE"
5050     PRINT:PRINT " (c) Olaf":PRINT"     Hartwig"
5060     PRINT:PRINT "     1986"
5070     PRINT: PRINT
5080     PRINT" Ausgang      gefunden! "
5090     PRINT:PRINT durchgang;" Durchgaenge"
5100 :
5110     PRINT: PRINT: PRINT" Druecken Sie":PRINT " eine Taste..."
5120     q$=INKEY$:IF q$="" THEN 5120
5200 RETURN
```

4.6 Ein Fazit des Micro-Mouse-Contest

An den vorgestellten Micro-Mouse-Simulationsbeispielen wird sehr gut deutlich, was »intelligente« Roboter ausmacht: entscheidend ist die Flexibilität der Software.
Wir haben aber andere Faktoren, die im Micro-Mouse-Contest mit über Sieg und Niederlage entscheiden, noch nicht berücksichtigt. Es sind vor allem die Sensortechniken sowie die Mechanik. In der Praxis der letzten Micro-Mouse-Wettbewerbe hat sich gezeigt, daß die Software der unterschiedlichen Mäuse seit 1981 fast standardisiert ist und jeweils ein ähnliches Lösungskonzept verfolgt.
Heute ist das Hauptanliegen des Wettbewerbs die Verbesserung der Sensor- und Steuerungstechnik, um aus den gewonnenen Ergebnissen Erkenntnisse für Industrie-Roboter ableiten zu können.

4.6.1 Die Micro-Mouse-Sensortechnik

Die verwendeten Sensortechniken sind recht vielfältig. Kaum eine Maus gleicht einer anderen. Die eingesetzten Techniken reichen von ausgeklügelten mechanischen Sensoren über Ultraschall-Sensoren bis hin zu optischen, wie z.B. Infrarot-Sensoren. Teilweise werden auch einzelne Sensorentypen miteinander gekoppelt.
Denkbar ist jedoch auch eine Steuerung über moderne Laser-Sensoren. Mehr über den Einsatz der Lasertechnologie in Verbindung mit Robotics erfahren Sie in Abschnitt 4.8.4.
Die Verbesserungen und Verfeinerungen der Sensorentwicklung zielen vor allem darauf ab, die unmittelbare Umgebung der Maus näher und noch detaillierter zu erfassen. Dadurch sollen Koor-

dinationsprobleme der Maus vermieden werden. So muß der Micro-Mouse-Roboter beispielsweise ständig rechtwinklig zu den Labyrinthwänden stehen, um einen Weg durch den Irrgarten zu finden.
Schwieriger als dieses Problem ist jedoch die genaue Erfassung der begehbaren Abzweigungen vom momentanen Labyrinthweg, auf dem sich die Maus befindet. Auch auf dem letzten Micro-Mouse-Contest fielen Mäuse aufgrund von Fehlern in der Sensortechnik aus.

4.6.2 Die Micro-Mouse-Mechanik

Auf dem Gebiet der Mechanik hat der Micro-Mouse-Wettbewerb die größten Erfolge hervorgebracht. Die Aufgabenstellung der Mechanik läßt sich grob in zwei Teilbereiche untergliedern:

Die Geschwindigkeit

Da ist zum einen die Schnelligkeit mechanischer Vorgänge. Geschwindigkeit ist Trumpf, da der Wettbewerb gegen die Uhr ausgetragen wird. Heute rasen die Mäuse mit hoher Geschwindigkeit durch das Labyrinth. Sie sind sehr wendig, können also in kürzester Zeit Richtungsänderungen vornehmen.
Die Entwicklung verlief parallel zur Verbesserung der Sensortechnik. Um derartige Geschwindigkeiten erreichen zu können, ist es natürlich für die Sensoren des Micro-Mouse-Roboters notwendig, innerhalb von Sekundenbruchteilen die Umgebung der Maus richtig und vor allem umfassend zu erfassen.
Eine weitere Geschwindigkeitsverbesserung liegt in Faktoren, die sich mit denen aus dem Automobilrennsport vergleichen lassen. Einer der Faktoren ist beispielsweise die maximale Beschleunigung des Roboters nach einer Richtungsänderung oder aus dem Stand. Genauso wichtig ist das Erreichen einer Höchstgeschwindigkeit in den schmalen Labyrinthgängen, ohne daß die Maus entgleist.

Das Konzept der Economical-Mouse

Die zuvor beschriebenen Faktoren führten im letzten Entwicklungsstadium zur sogenannten Economical-Mouse. Sie wurde von Alan Dibley aus England 1982 eingeführt.
Er erstand den billigsten Mikrocomputer, entfernte das Gehäuse und die Tastatur und entwickelte eine aus leichtem Balsaholz aufgebaute Maus. Sie wurde von leistungsfähigen Sensoren und Servo-Motoren mit großer Beschleunigungsfähigkeit gesteuert. Mit diesem Konzept erreichte er sehr gute Erfolge im Wettbewerb.

4.7 Übertragung der Micro-Mouse-Ergebnisse auf Industrie-Roboter

Wie lassen sich die gewonnenen Erfahrungen und Ergebnisse auf Industrie-Roboter anwenden?

4.7.1 Die Sensortechnik

Sensoren werden für kommerzielle Roboter immer wichtiger. Mit steigendem Maß an Eigenintelligenz und Flexibilität, die einem Roboter abverlangt werden, steigt auch der Bedarf an präzisen Sensoren. Und hier lassen sich die Ergebnisse der Micro-Mouse-Sensoren direkt übertragen. Neben anfangs nur einfachen mechanischen Sensoren spielen Infrarot, Ultraschall und neuerdings auch Laser-Sensoren eine immer größere Rolle.
Die Sensoren ermöglichen es den Robotern, die Umwelt näher und genauer wahrzunehmen und sich entsprechend zu verhalten.

4.7.2 Die Mechanik der Industrie-Roboter

Hier wurden vor allem die Präzision und die mechanische Geschwindigkeit der Micro-Mäuse übernommen.
Wichtig ist dabei vor allem das Feedback einzelner Servos. Der Steuerungs-Computer des Roboters muß jederzeit genau wissen, in welcher Stellung sich ein Servo-Gelenk des Roboters befindet. Entscheidend ist hier die Genauigkeit.
Sehr nützliche Anstöße für die Industrie-Roboter liefert vor allem die Geschwindigkeit der Mechanik. Je schneller ein Roboter einen Bewegungsvorgang ausführen kann, desto effektiver und leistungsfähiger läßt er sich einsetzen.

4.7.3 Der Micro-Cat-Wettbewerb

Der Micro-Mouse-Wettbewerb wurde 1980 von John Billingsley ins Leben gerufen. Heute liefert er allerdings nur noch wenige Anstöße zur Entwicklung neuer Techniken. Vor allem deshalb wurde der Euro-Mouse-Wettbewerb, als er von den Japanern übernommen wurde, wesentlich erweitert, um so weitere Erkenntnisse für Roboter zu gewinnen.
Die japanische »Science Foundation« veranstaltet jährlich einen Roboter-Wettbewerb, der wesentlich höhere Anforderungen an die Software, die Sensoren und die Mechanik stellt. Neben den Micro-Mäusen starten dort nämlich auch Micro-Cats. Die wesentlich komplizierter aufgebauten Roboter können sich in drei Dimensionen bewegen. Eine weitere Besonderheit besteht darin, daß das zu bewältigende Labyrinth im Gegensatz zum Euro-Mouse-Irrgarten Kreuzungen enthält.
Es ist leicht einsehbar, daß die Micro-Cats noch ausgeklügeltere Sensoren als die Mäuse benötigen, um sich im Labyrinth zurechtzufinden.

Doch auch die so erzielten Ergebnisse reichen heute nicht mehr aus. Deshalb wurde kürzlich ein neuer, weltweiter Roboter-Wettbewerb vorgeschlagen.

4.7.4 Robot Ping-Pong

Ein Tischtennis-Wettbewerb zwischen zwei Robotern!
Dieser Wettbewerb wurde vom Gründer und Organisator des Micro-Mouse-Contest entwickelt. John Billingsley hat dabei das Ziel vor Augen, die Entwicklung von Robotern in eine neue Dimension zu führen.
Denken Sie einmal kurz darüber nach, was ein Ping-Pong spielender Roboter zu leisten hat. Die größte Schwierigkeit liegt in der sehr hohen Geschwindigkeit des Ping-Pong-Balles. Die Mechanik muß um Zehnerpotenzen schneller als die der Micro-Mäuse sein, um den Ball zu treffen.
Damit verbunden steigt die Komplexität der Software. Berücksichtigt werden müssen kompliziertteste Faktoren wie die Ballgeschwindigkeit, der Aufschlagswinkel und später auch angeschnittene Bälle.
Die umfangreichsten Anforderungen werden aber an die Sensoren gestellt. Ein Ping-Pong-Match läßt sich nur über ein Visions-System durchführen. Auf dem Gebiet werden dann wohl auch die größten Fortschritte zu erwarten sein. Für die kommerzielle Industrie ist das insofern sehr interessant, als sehende Computer und Roboter im Bereich der Robotics einen immer größeren Stellenwert einnehmen. Mehr darüber später in diesem Kapitel.
Der Wettbewerb ist nicht ganz so futuristisch, wie es zunächst erscheinen mag. Das wird deutlich, wenn man sich die Wettbewerbsbedingungen genauer anschaut. Die Tischtennisplatte ist zwei Meter lang und einen halben Meter breit. Über dem Netz, das 25 cm hoch ist, befindet sich in 50 cm Höhe ein Rahmen, der die maximale Höhe des fliegenden Balles einschränkt. Die Einschränkungen sind erforderlich, damit die heute schnellste Mechanik in der Lage ist, den Ball zu erreichen.
Nach John Billingsley ist von der mechanischen Seite her lediglich ein umgebauter sehr schneller X/Y-Plotter notwendig.
Entscheidend für die Effektivität eines Ping-Pong-Roboters wird bei den vorgegebenen Rahmenbedingungen zum einen das optische System sein, das die Bahn des Balles in jedem Moment genau verfolgen muß. Genauso wichtig wird, wie durch erste Simulationen festgestellt wurde, die Spielstrategie sein. Sie muß so intelligent wie möglich sein. Um einen für den Gegner schwierigen Return zu spielen, ist größte Präzision des Roboters notwendig.
Wir werden in nächster Zukunft sicher mehr über den Wettbewerb hören.

4.7.5 Intelligente Roboter der fünften Generation

Werfen wir nun einen Blick auf moderne Industrie-Roboter sowie Forschungs-Roboter.
Wie der Titel bereits andeutet, haben die Japaner in ihr Projekt intelligenter Computer der fünften Generation auch ein nationales Roboter-Projekt mit einbezogen. Das zeigt den sehr hohen Stellen-

wert, der Robotics innerhalb der Künstlichen Intelligenz zugewiesen wird. Der entscheidende Schwerpunkt, den die Japaner sich gesetzt haben, ist dabei die Entwicklung leistungsfähiger, möglichst universell anwendbarer Visions-Systeme.
Das angestrebte Bilderkennungs-System soll in der Lage sein, schließlich rund 100 000 Bilder speichern und zum Vergleich mit neuen Bildern heranziehen zu können.
Die Entwicklung läuft dabei in drei Schritten ab:
Der erste Schritt ist die Versuchsphase. In ihr werden Probleme wie die Hardware-Architektur, die Wahl von Analysemerkmalen (etwa die Bestimmung von Grenzen im Bild), die Generierung von Displays sowie die Organisation von Bilddatenbanken gelöst.
In der zweiten Phase sollen leistungsfähige Pilotmodelle entwickelt werden.
In der letzten Phase schließlich wird das Visions-System mit anderen Projekten der fünften Generation zusammengesetzt. In diese Phase fällt dann auch die Integration des Systems in Roboter.
Allerdings wird das Projekt frühestens in den nächsten fünf Jahren realisiert werden. Robotics nimmt aber im Bereich der KI und der kommerziellen Anwendung bereits heute einen immer größeren Stellenwert ein.

4.8 Der Stellenwert von Robotics

Das Geschäft mit Robotertechnologie ist »Big Business«. Allein in den USA wurden im letzten Jahr 300 Millionen Dollar in diesem Geschäftsbereich erwirtschaftet. Die Wachstumsrate lag bei 25 Prozent.
Derzeit sind zwar noch 260 Firmen im Robotics-Bereich tätig, doch es wird angenommen, daß drei von vier in den nächsten Jahren verschwinden werden.

4.8.1 Die Intelligenz kommerzieller Roboter

Bis vor wenigen Jahren waren Roboter noch nicht intelligent. Viele Roboter waren auch keine wirklichen Roboter, sondern einfache Handhabungsautomaten. Doch in den letzten Jahren führten vor allem Visions-Systeme zu einem gehörigen Maß an Eigenintelligenz. Sie ist vor allem durch eine größere Eigenständigkeit der Software der Systeme bedingt.
Moderne intelligente Roboter können beispielsweise über eine Kamera fehlerhafte Werkstücke entdecken und aussortierten. Auch wenn Bausteine nicht wie üblich nebeneinander auf einem Fließband, sondern aufeinander liegen, kann der Roboter sie wieder in die richtige Position einordnen.
Ein wesentlicher Vorteil intelligenter sehender Roboter besteht auch darin, daß sie in der Lage sind, aus einer großen Anzahl unterschiedlichster Werkstücke das derzeit gerade benötigte herauszusuchen.

Ein weiteres wichtiges Einsatzgebiet für mit Visions-Systemen ausgestattete Roboter ist die Montage. Sehende Roboter übernehmen beispielsweise schon heute die Justierung und Montage von Rädern an Autos. Die Schraublöcher beider zu verbindenden Teile werden gesehen und das Rad entsprechend von Robotern ausgerichtet und anschließend montiert.
Als Augen eines Roboters werden heute nicht mehr die üblichen Röhrenkameras eingesetzt. Statt dessen findet man fast ausschließlich MOS-Halbleiterkameras, die zum einen wesentlich schneller als Röhrenkameras, zum anderen kompatibel zum Computer sind.

4.8.2 Aktuelle Robotics-Trends

Der Trend in Robotics läuft darauf hinaus, daß Visions-Systeme einen immer größeren Stellenwert in Robotics einnehmen werden.
Außer den vorgestellten Anwendungsbereichen der Montage sowie der Selektion spezifischer Werkstücke wird die Qualitätssicherung und Überprüfung stärker in den Vordergrund rücken. Bei vielen Produkten ist heute der Aufwand für die Qualitätssicherung und -prüfung vergleichbar mit den Herstellungskosten. Automatische Prüfverfahren durch sehende Roboter ermöglichen eine objektive, ermüdungsfreie und rationelle Qualitätskontrolle. So ist es heute bereits möglich, mittels visueller Prüfverfahren beispielsweise die Oberflächenqualität eines Werkstücks zu bestimmen und so auch kleine Risse zu erkennen.
Weitere Schwerpunkte der aktuellen Robotics-Forschung deuten sich heute bereits an:
- Robotergerechtes Konstruieren (CAD/CAM)
- Graphische Simulation
- Mensch-Maschine-Schnittstelle
- Expertensysteme und Robotics
- Anwendungsbereich Raumfahrt
- Roboter in Haushalts- und Dienstleistungsbereichen
- Optimierung der Rechner-Architektur
- Einsatz von Laser-Technologien

4.8.3 Das US-Air-Force-Robotics-Programm

Der weltweite Führer in Robotics-Forschungen ist nicht die kommerzielle Industrie, sondern die US Air Force. Die Air Force hat das Ziel, über flexible Automation die Produktivität und Effektivität ihrer militärischen Flugzeuge und deren Systeme zu verbessern. Die derzeitigen Air-Force-Projekte werden daher große Auswirkungen auf die gesamte Robotics-Entwicklung der Zukunft haben.
Schwerpunkte des Programms sind Visions-Systeme mit der Fähigkeit zur »Real Time Vision«. Dazu werden Real-Time-Sensoren, Kontrollsysteme sowie Systemintegration und die entsprechende Hardware entwickelt. Die KI-Software entsteht derzeit im weltweit bekannten KI-Labor der Stanford University, das von Prof. E. Feigenbaum geleitet wird.

Das Programm verläuft in zwei Phasen:
In der ersten Phase wird ein »micro-manipulator« mit integrierten »force-sensing«-Fingern sowie ein »Real-Time«-Kontrollsystem mit Künstlicher Intelligenz sowie einem fortschrittlichen »Output«-System entwickelt.
Die zweite Phase beinhaltet die Konstruktion eines Roboter-Systems, das sowohl zufällig angeordnete Werkstücke als auch kompliziertere, d.h. komplexere Zubehörteile für Flugzeuge zusammensetzen kann.
Die neuartige Grundidee dabei ist, von der bisherigen sorgfältig auf den Roboter abgestimmten Umgebung abzukommen. Der Roboter wird also beispielsweise ohne ein Fließband operieren können und bewegt sich selbst zu einzelnen Werkstücken. Neu ist auch, daß Einschränkungen bezüglich der Lichtverhältnisse, der Geschwindigkeit sowie der Genauigkeit heutiger Visions-Systeme beseitigt werden sollen.
Das Hauptziel ist also die Entwicklung flexibler intelligenter Fertigungs-Systeme, die in »Real-Time« operieren können.

4.8.4 Robotics und Laser-Technologie

Die moderne Laser-Technologie findet immer stärkeres Interesse bei Robotics-Wissenschaftlern. Erste Einsatzbereiche sind bereits erschlossen. Man unterscheidet zwei Arten von Lasereinsätzen in Robotics: High- und Low-Energy-Laser.
High-Energy-Laser, wie sie derzeit im amerikanischen »Star Wars«-Weltraum-Verteidigungsprogramm eingesetzt werden, dienen in Verbindung mit Robotics beispielsweise zur Materialbearbeitung. High-Energy-Laser können dicke Stahlplatten in kurzer Zeit mit größter Präzision zerschneiden.
Low-Energy-Laser lassen sich dagegen ideal in der Sensor-Technologie einsetzen. Sie können mit großer Genauigkeit Entfernungen messen und so beispielsweise dem Feedback einzelner Roboterelemente dienen.
Die Laser-Technologie ermöglicht insgesamt gesehen interessante neuartige Anwendungsbereiche für Roboter.

5 Experten-Systeme

5.1 Stellenwert von Experten-Systemen

Das Gebiet der Experten-Systeme hat bis heute die für die praktische Anwendung wichtigsten Resultate erbracht. Während die meisten der KI-Programme der anderen KI-Gebiete wie Spracherkennung, Deduktionssysteme, Wissensverarbeitung und zu einem Teil auch Bildverarbeitung noch nicht praktisch für jeden Anwender oder in der Wirtschaft einsatzbereit sind, finden fertige Experten-Systeme und Experten-System-Shells (Experten-Systeme ohne Wissensbasis) eine immer stärkere Verbreitung.
Die große Popularität wird verständlich, wenn man sich verdeutlicht, was Experten-Systeme heute bereits leisten können. Ein in der KI-Geschichte wichtiges Experten-System ist MYCIN. Es handelt sich dabei um ein rein medizinisches Diagnose-Experten-System. MYCIN diagnostiziert Blut-Infektionen und berät den behandelnden Arzt über die effizientesten Therapiemöglichkeiten. Experten-Systeme werden als »Intelligente Assistenten« bezeichnet. Auch MYCIN übt als Assistent eine Beraterfunktion aus. Seine Wissensbank beinhaltet das Wissen zahlreicher Experten für Blut-Infektionen. Zur Fertigstellung des Regelmechanismus und der Wissensbank mußte sehr viel Zeit und Mühe investiert werden, aber die Leistungsfähigkeit von MYCIN ist auch erstaunlich. Bei den Diagnosevorschlägen wird die Stufe von menschlichen Experten des speziellen Fachgebiets von Blut-Infektionen erreicht. Die Diagnosen auf allgemeinem Gebiet tätiger Ärzte werden durch MYCIN sogar noch übertroffen.
Damit das System ständig kontrolliert werden kann, verfügt es über eine Erklärungskomponente. Der Arzt kann MYCIN jederzeit fragen, warum es einen speziellen Diagnosevorschlag unterbreitet hat, warum es bestimmte Folgerungswege nicht beschritten hat oder wie es auf die einzelnen Schlußfolgerungen gekommen ist.
Was MYCIN zu seiner Diagnose benötigt, sind lediglich alle wissenschaftlichen Außendaten, wie Ergebnisse von Laboruntersuchungen und die Krankheitsgeschichte des Patienten. Zusätzlich führt es wie fast alle Experten-Systeme, während es die Diagnose erstellt, einen Dialog mit dem jeweiligen Arzt.

Nachdem sich MYCIN in der Praxis ausgezeichnet bewährt hatte, wurde es zu einer Experten-System-Shell umgewandelt. Man hat, um die Erstellung neuer Experten-Systeme für andere Problemgebiete zu vereinfachen, das Folgerungssystem von der Wissensbank getrennt. Das MYCIN-Folgerungssystem wurde also isoliert und als Shell verwendet. Dieses leere Experten-System kann mit beliebigen Wissensbänken kombiniert werden. Um das System auf neue Problemgebiete abzustimmen, müssen die KI-Wissensingenieure sich nur noch um die Erstellung der neuen Wissensbank für das Gebiet kümmern und können das alte Folgerungssystem weiterverwenden. Eine praktische Anwendung der MYCIN-Shell liegt auf dem Gebiet der Lungenkrankheiten. Das neue Experten-System hat sich bereits in der Praxis bewährt.
Medizinische Experten-Systeme stellen derzeit die größte Gruppe unter den Experten-Systemen dar. Das Einsatzgebiet von Experten-Systemen ist aber schon heute sehr vielfältig: Es gibt intelligente Geologenmaschinen, wie PIPMETER, das nach Erdöl sucht, militärische Experten-Systeme, Betriebswirtschafts-Systeme, Experten-Systeme in der Biotechnik, der Chemie, spezielle Systeme für Computeranlagen, Erziehung, Rechtswissenschaften und Systeme für technische Probleme, wie die Entwicklung dreidimensionaler mikroelektronischer Schaltungen.

5.2 Experten-Systeme auf dem CPC

Im folgenden werden wir unterschiedliche Ansätze zur Entwicklung von Experten-Systemen auf Home-Computern in BASIC untersuchen.

5.2.1 Grundlagen von Experten-Systemen

Was ist eigentlich genau ein menschlicher Experte? Und was muß ein Experten-System leisten können?

Ein menschlicher Experte ist jemand, der viel über ein bestimmtes Fachgebiet weiß. Er kann ein sinnvolles Urteil über Probleme seines Gebietes abgeben. Die Fähigkeiten mußte er sich in jahrelangem Training, in Schulungen und durch Sammeln von Erfahrungen aneignen.

Es ist verständlich, daß fähige Experten knapp und für die Industrie recht teuer sind. U.a. aus diesem Grund wurden die oben beschriebenen Experten-Systeme entwickelt.
Ein menschlicher Experte analysiert und löst ein spezifisches Problem im wesentlichen folgendermaßen:

1. Zuerst sammelt er alle verfügbaren Informationen zu der zu lösenden Problemstellung seines Fachgebietes.
2. Dann vergleicht er die Informationen, die er zu der Problemlösung erhielt, mit dem Wissen, das er sich während seiner Schulung und durch Erfahrung angeeignet hat. Dabei versucht er, einen Zusammenhang zwischen beiden Informationen zu finden.
3. Hat er einen derartigen Zusammenhang gefunden, so gibt er eine entsprechende Meldung über die Problemlösung aus. Konnte er keine Beziehung zwischen seinem Wissen und der Problemlösungsaufgabe finden, so teilt er das ebenfalls mit.

5.2.2 Realisierung eines einfachen Database-Programms

Die Aufgabe des oben beschriebenen Lösens von Problemen kann im wesentlichen von einem einfachen Database-Programm gelöst werden, das die Informationen einer Aufgabenstellung mit dem in der Database gespeicherten Wissen vergleicht.

Angenommen, wir stehen vor dem Problem, verschiedene Laute und Geräusche unterschiedlichen Tieren zuzuordnen. Dieses Expertenproblem ist verhältnismäßig einfach zu lösen. Das Programm »DATABASE« fragt zuerst nach Informationen zum Problem, in diesem Fall nach dem Geräusch eines Tieres. Anschließend wird dieses mit den Informationen der Database verglichen. Danach gibt das Programm aus, ob ein Zusammenhang gefunden wurde oder nicht.

Unten sehen Sie das Listing des Programms »DATABASE« in der Version 1:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: Database Version 1
5 :
6 MODE 1
10 :
90 REM ************ Supervisor **************
100       GOSUB 1000
110           REM Initialisation
200       GOSUB 2000
210           REM Vergleich
300       GOSUB 3000
310           REM Ausgabe
320 :
330       PRINT
400       PRINT "Wuenschen Sie noch einen Durchgang?(J/ )"
420       a$=INKEY$
430         IF a$="" THEN 420
440         IF a$="j" THEN 200
450 STOP
460 :
1000 REM ********* Initialisation ***********
1100     DIM l$(20)
1110         REM linkes Feld
1120     DIM r$(20)
1130         REM rechtes Database Feld
1140 :
1200      FOR f=1 TO 10
1210          READ l$(f)
1220      NEXT f
1230 :
1300      FOR f=1 TO 10
1310          READ r$(f)
1320      NEXT f
1330 :
1340 RETURN
1350 :
1400 REM ----------- Datas Tiere ------------
1410          DATA Lerche
1420          DATA Hahn
1430          DATA Kuh
```

```
1440          DATA Schaf
1450          DATA Hund
1460          DATA Katze
1470          DATA Schwein
1480          DATA Huhn
1490          DATA Frosch
1500          DATA Eule
1510 :
1520 REM ---------- Datas Geraeusche --------
1530          DATA zwitscher
1540          DATA kiekerikie
1550          DATA muh
1560          DATA maeh
1570          DATA wau
1580          DATA miau
1590          DATA grunz
1600          DATA gluck
1610          DATA quack
1620          DATA uhu
1630 :
2000 REM ****** Vergleich der Database ******
2030 :
2050  PRINT
2060  INPUT "Welches Geraeusch macht das Tier         ";ge$
2070  PRINT
2080 :
2090 ma=0
2095   REM marker reset
2100 FOR f=1 TO 10
2110    IF r$(f)<>ge$ THEN NEXT f:ma=66:RETURN
2120 RETURN
2130    REM beziehung gefunden
2990 :
3000 REM ******* Ausgabe des Resultats ******
3100 :
3120   PRINT "Das wahrgenommene Geraeusch ";ge$
3130   PRINT "gehoert zu dem Tier ";l$(f)
3150 RETURN
```

Für unsere Aufgabenstellung genügt das obige Programm. Es hat aber den Nachteil, zu unflexibel zu sein. Bei vielen Expertenaufgaben besteht nicht nur der hier realisierte Zusammenhang zwischen jeweils einem Geräusch und je nur einem zugehörigen Tier. Vielmehr sind die Beziehungen wesentlich vielfältiger.

Auch in unserem Beispiel gibt es schon die Möglichkeit von Mehrfachzuweisungen. So können dem Vogel »Lerche« alle folgenden Geräusche zugewiesen werden:

singen, zwitschern, piepen, trällern

und so weiter. Die Mehrfachbedeutungen lassen sich durch eine Änderung der Database in unser Programm implementieren.

Das neue Programm »DATABASE« sieht dann folgendermaßen aus:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: Database Version 2
5 :
6   MODE 1
7   PAPER 0:PEN 1
8   BORDER 12
9   WIDTH 60
10 :
90 REM ************ Supervisor **************
100      GOSUB 1000
110          REM Initialisation
200      GOSUB 2000
210          REM Vergleich
300      GOSUB 3000
310          REM Ausgabe
320 :
330      PRINT
400      PRINT " Wuenschen Sie noch einen Durchgang? (J/_) ";
420      a$=INKEY$
430        IF a$="" THEN 420
440        IF a$="j" THEN PRINT a$:GOTO 200
450 STOP
460 :
1000 REM ********* Initialisation ***********
1100    DIM l$(20)
1110       REM linkes Feld
1120    DIM r$(20)
1130       REM rechtes Database Feld
1140 :
1150    WINDOW 1,30,1,6:PAPER 1:PEN 0:CLS
1160    PRINT:PRINT" Verbesserte KI-Demo-DATABASE"
1170    PRINT:PRINT TAB(9);"(c) Olaf Hartwig"
1180    PRINT TAB(26);"1985"
1185    WINDOW 4,39,6,24:PAPER 2:CLS
1190 :
1200      FOR f=1 TO 13
1210          READ l$(f)
1220      NEXT f
1230 :
1300      FOR f=1 TO 13
1310          READ r$(f)
1320      NEXT f
1330 :
1340 RETURN
1350 :
1400 REM ----------- Datas Tiere ------------
1420          DATA Hahn
1430          DATA Kuh
1440          DATA Schaf
1450          DATA Hund
1460          DATA Katze
1470          DATA Schwein
1480          DATA Huhn
1490          DATA Frosch
1500          DATA Eule
1502          DATA Lerche
```

```
1504         DATA Lerche
1506         DATA Lerche
1508         DATA Lerche
1510 :
1520 REM ---------- Datas Geraeusche --------
1540         DATA kiekerikie
1550         DATA muh
1560         DATA maeh
1570         DATA wau
1580         DATA miau
1590         DATA grunz
1600         DATA gluck
1610         DATA quack
1620         DATA uhu
1622         DATA piep
1624         DATA traellert
1626         DATA zwitschert
1628         DATA singt
1630 :
2000 REM ****** Vergleich der Database ******
2030 :
2050  PRINT
2060  INPUT " Welches Geraeusch macht das Tier    _>";ge$
2070  PRINT
2080 :
2090 ma=0
2095   REM marker reset
2100 FOR f=1 TO 10
2110    IF r$(f)<>ge$ THEN NEXT f:ma=66:RETURN
2120 RETURN
2130    REM beziehung gefunden
2990 :
3000 REM ******* Ausgabe des Resultats ******
3100 :
3105 IF ma=66 THEN 4000
3110 :
3120   PRINT " Das wahrgenommene Geraeusch ";ge$
3130   PRINT " gehoert zu dem Tier "; l$(f)
3150 RETURN
3200 :
4000   PRINT " Das Geraueesch ist noch nicht       bekannt!"
4100 RETURN
```

Das Programm löst unsere Problemstellung ganz zufriedenstellend.
Ein Beispiel-Programmablauf ist im folgenden dargestellt.

<u>Verbesserte KI-Database</u>
Welches Geraeusch macht das Tier?
?maeh

Das wahrgenommene Geraeusch maeh
gehoert zu dem Tier schaf

Wuenschen Sie noch einen Durchgang?
(J/_) j

Welches Geraeusch macht das Tier?
?
.
.
.

Das Programm ist in seiner Leistungsfähigkeit immer noch recht begrenzt, obwohl es unser Problem gut löst. Es wird jeweils nur eine Frage gestellt und das Programm gibt nur eine korrekte Antwort.
Zur Lösung komplexerer Probleme reicht »DATABASE« nicht aus. Daher entwickeln wir im folgenden einen neuen Lösungsansatz.

5.2.3 Das EXEC-Programm

Dieses Experten-System löst ein ähnliches, aber komplexeres Problem als das Programm DATABASE aus dem Gebiet der Biologie, jedoch nach einem anderen Verfahren.
Unsere Problemstellung ist hier, verschiedene Lebewesen anhand individueller Fragen zu erkennen. Schauen Sie sich einmal die prinzipielle Lösungsstrategie von EXEC an. Sie macht das Experten-System EXEC deutlich universeller als das Programm DATABASE.
Das EXEC-Listing lautet:

```
1 REM Kuenstliche Intelligenz auf dem CPC
4 REM Titel: EXEC
5 :
6    MODE 1
7    PAPER 0:PEN 1:CLS
8    WIDTH 60
9    BORDER 9
10 :
90 REM *************** Supervisor *****************
100       GOSUB 1000
110           REM Initialisation
200       GOSUB 2000
210           REM Vergleich
300       GOSUB 3000
310           REM Ausgabe
320 :
330       PRINT
400       PRINT " Wuenschen Sie noch einen  Durchgang?(J/_)"
420       a$=INKEY$
430        IF a$="" THEN 420
440        IF a$="j" THEN RUN
450 STOP
460 :
1000 REM *********** Initialisation **************
1002  WINDOW 8,39,8,24:PAPER 2:CLS
```

```
1010  WINDOW 3,10,1,20
1020    PAPER 1:PEN 0:CLS
1050      PRINT: PRINT " EXEC"
1060      PRINT: PRINT: PRINT" (C)"
1070      PRINT " Olaf"," Hartwig"
1075      PRINT " 1985"
1080        ORIGIN 45,150:DRAW 1,250,0
1090        WINDOW 11,39,8,24
1092        PAPER 2:CLS
1095 :
1100      c=0
1110      REM Zuweisungsindikator
1200 RETURN
1210 :
2000 REM ************ Hauptprogramm **************
2010   PRINT "Bitte denken Sie nun an eines";
2020   PRINT "der 4 Lebewesen Hund, Mensch,";
2030   PRINT "Delphin oder Adler ..."
2035   PRINT
2040 :
2100,     PRINT"Hat das Lebewesen zwei Beine?";
2110         GOSUB 2500
2120      PRINT"Kann es gehen?"
2130         GOSUB 2500
2140      PRINT"Kann es fliegen?"
2150         GOSUB 2500
2160 :
2170 RETURN
2300 :
2500 REM ---------- Antwortensubroutine ----------
2510    PRINT "Druecken Sie (J)a oder (N)ein";
2515    PRINT
2520     FOR f=1 TO 50
2530     NEXT f
2540   b$=INKEY$
2550   IF b$="" THEN 2540
2600  IF b$="j" THEN c=c+1:RETURN
2610  IF b$<>"n" THEN 2540
2615 :
2620 RETURN
2990 :
3000 REM ************ Ergebnisausgabe ************
3002   WINDOW 15,40,18,25: PAPER 3: PEN 1: CLS
3005    c=c+1
3010      ON c GOSUB 3050,3100,3200,3300
3020      GOTO 3500
3030 :
3050      REM c=1
3060         t$="Delphin"
3070      RETURN
3080 :
3100      REM c=2
3110         t$="Hund"
3120      RETURN
3130 :
3200      REM c=3
```

```
3210          t$="Mensch"
3220        RETURN
3230 :
3300        REM c=4
3310          t$="Adler"
3320        RETURN
3330 :
3500  PRINT
3510    PRINT" Das Lebewesen ist ein _>"
3520    PRINT " ";t$
3600 RETURN
```

Nach dem EXEC-Experten-System-Grundmuster können Sie beliebige Experten-Systeme für eigene Problemlösungen entwickeln. Im Programm werden jeweils unterschiedliche Parameter so verknüpft, daß ein einziger arithmetischer Wert, der sich aus den Verknüpfungen ergibt, für eine spezielle Antwort typisch ist. Der Wert wird durch die Variable »c« festgelegt.
Der Programmablauf ist im folgenden dargestellt:

EXEC (Olaf Hartwig '86)

Bitte denken Sie an eines
der vier Lebewesen Hund, Mensch,
Delphin oder Adler ...

Hat das Lebewesen zwei Beine?
Druecken Sie (J)a oder (N)ein... N

Kann es gehen?
Druecken Sie (J)a oder (N)ein... N

Kann es fliegen?
Druecken Sie (J)a oder (N)ein... N

Das Lebewesen ist ein.. delphin

Wuenschen Sie einen neuen Durchgang?
(J/_)

5.2.4 Ein Fazit

Wenn Sie sich die beiden Lösungsansätze anschauen, werden Sie trotz der recht effektiven Problemlösungs-Strategie, die beide Experten-Systeme verfolgen, einen wesentlichen Mangel feststellen: Beide Systeme können nicht im Dialog mit dem Benutzer von selbst hinzulernen. Außerdem haben sie nicht die Fähigkeit, fehlerhafte Daten selbst zu erkennen und zu löschen bzw. umzuändern.
Da die Fähigkeit des Lernens, also die Erweiterung der Wissensbank, die Fehlerkontrolle und die Möglichkeit, fehlerhafte Regeln zu korrigieren, für moderne Experten-Systeme eine entscheidende Voraussetzung sind, muß auch die Programmstruktur entsprechend geändert werden. In den

folgenden Abschnitten wird Ihnen daher das Programm EXPERT vorgestellt, das auf einer Suchbaumstruktur aufgebaut ist und die geforderten Fähigkeiten besitzt.

5.3 Das Experten-System EXPERT

Das EXPERT-System stellt die Grundstufe zahlreicher Experten-Systeme dar. Es läßt sich als Experten-System-Shell anwenden. Das bedeutet, daß es über eine Flexibilität verfügt, die es ermöglicht, unterschiedliche Wissensbänke zu verwalten und so unterschiedliche Aufgabenbereiche zu bewältigen.

Das Grundkonzept des Suchbaums ist altbewährt. Als Standard-Anwendung, anhand derer die Funktionsweise des Suchbaums am besten demonstriert werden kann, gilt das Evolutions-Experten-System. Geben Sie zunächst das EXPERT-Listing ein:

```
10 ' EXPERT (c) Olaf Hartwig '86
20 ' Initialisierung
30 GOSUB 3000
40 '
100 ' Expert-Hauptschleife
110 ' Marker fuer Baumebene Reset
115 CLS
120 baum=1: REM momentane Baumebene Reset
130 '
135 PRINT TAB(19)"Anzahl der gespeicherten Tiere: ";tiere
140 PRINT "EXPERT-Frage:        "; fr$(baum)
150 INPUT "ANWENDER-Antwort: ", antwort$
155 IF antwort$="ende" OR antwort$="stop" THEN PRINT:PRINT
    "Bis zum naechsten Mal!": END
160 IF LEN(antwort$)=0 OR LEN(antwort$) >4 THEN 150
165 antwort$=LEFT$(antwort$,1)
170 '
180 ' Verzweigung aufgrund der linken und rechten
190 ' Fragespalten und der Antworten
200 IF rechts(baum)=0 AND antwort$="j" OR antwort$="J" THEN 1000
210 IF rechts(baum)<0 AND antwort$="j" OR antwort$="J" THEN
    hilf=-rechts(baum): GOTO 2000
220 IF links(baum)=0 AND antwort$="n" OR antwort$="N" THEN 1000
230 IF links(baum)<0 AND antwort$="n" OR antwort$="N" THEN
    hilf=-links(baum): GOTO 2000
240 IF antwort$="n" OR antwort$="N" THEN baum=links(baum)
250 IF antwort$="j" OR antwort$="J" THEN baum=rechts(baum)
260 ' Schleifenende
270 GOTO 140
990 '
1000 ' Modul fuer nicht bekannte Tiere
1020 PRINT
1030 PRINT "Das gedachte Tier ist noch unbekannt!"
1040 PRINT "Bitte geben Sie naehere Informationen ein:"
1045 tiere=tiere+1
```

```
1050 INPUT "Wie lautet sein Name? ",na$(tiere)
1055 IF LEN(na$(tiere))<3 THEN 1050
1057 frage=frage+1
1060  INPUT "Mit welcher Frage kann man das Tier bestimmen? ",
      fr$(frage)
1070 IF LEN(fr$(frage))<6 THEN 1060
1080 IF antwort$="j" OR antwort$="J" THEN rechts(baum)=antwort+1
1090 antwort=antwort+1
1100 rechts(antwort)=-tiere
1110 links(antwort)=0
1115 IF antwort$="n" OR antwort$="N" THEN links(baum)=antwort
1120 INPUT "Lautet die Antwort auf die Frage ja oder nein? ",
     antwort$
1125 IF LEN(antwort$)<1 THEN 1120
1130 IF antwort$="n" OR antwort$="N" THEN links(antwort)=-tiere:
     rechts(antwort)=0
1140 GOTO 100
1200 '
2000 ' Modul fuer bekannte Tiere
2010 PRINT"Handelt es sich bei dem gesuchten Tier um ";
2020 PRINT na$(hilf);
2025 INPUT awort$: IF LEN(awort$)<1 THEN 2025
2030 IF awort$="j" OR awort$="J" THEN PRINT "Das Tier ist
     identifiziert!" ELSE GOTO 2100
2035 antwort$=""
2040 antwort$=INKEY$: IF antwort$="" THEN 2040
2050 GOTO 100
2100 ' Das Tier ist nicht bekannt
2105 PRINT
2110 PRINT"Das Tier ist noch unbekannt!"
2115 tiere=tiere+1
2120 INPUT "Wie lautet sein Name? ", na$(tiere)
2125 frage=frage+1
2130  PRINT "Mit welcher Frage kann man es von ";na$(hilf);
      " unterscheiden? "
2140 INPUT fr$(frage): IF LEN(fr$(frage))<5 THEN 2140
2145 antwort=antwort+1
2150 IF antwort$="j" OR antwort$="J" THEN rechts(baum)=frage
2160 rechts(antwort)=-tiere: links(antwort)=-hilf
2165 IF antwort$="n" OR antwort$="N" THEN links(baum)=-hilf
2170 INPUT "Lautet die Antwort auf die Frage ja oder nein? ",
     antwort$
2175 IF LEN(antwort$)<1 THEN 2170
2180 IF antwort$="n" THEN links(antwort)=-tiere:rechts
     (antwort)=-hilf
2190 GOTO 100
2200 :
3000 ' Initialisierung
3010 ' Datenfelder dimensionieren
3020 DIM fr$(200): 'Fragenfeld
3030 DIM na$(200): 'Namenfeld
3040 DIM links(200): 'linke Spalte
3050 DIM rechts(200): 'rechte Spalte
3060 ' Daten initialisieren
3070 frage=1
3075 fr$(frage)="Kann das zu suchende Tier schwimmen?"
```

```
3080 antwort=1
3090 ' Textwindow und Titel setzen
3100 MODE 2
3105 PAPER 0: PEN 1:CLS
3110 PRINT:PRINT TAB(25)"E X P E R T E N S Y S T E M"
3120 PRINT:PRINT TAB(45)"(c) Olaf Hartwig '86"
3125 PLOT 18,325: DRAW 600,325
3126 DRAW 600,390: DRAW 18,390
3127 DRAW 18,325
3130 WINDOW 3,75,7,23
3140 PAPER 1:PEN 0: CLS
3150 RETURN
```

5.3.1 Die Funktionsweise des Suchbaums

Im Anfangszustand ist der Suchbaum - bis auf die Eingangsfrage »Kann das zu suchende Tier schwimmen?« nicht mit Daten belegt. Erst während des Dialogs mit dem Anwender, also mit Ihnen, werden die Daten vom System gelernt. Ein Dialog kann zu Anfang folgendermaßen aussehen:

EXPERTEN-SYSTEM
(c) Olaf Hartwig '86

Anzahl der gespeicherten Tiere: 0
EXPERT-Frage: Kann das zu suchende Tier schwimmen?
ANWENDER-Antwort: ja

Das gedachte Tier ist noch unbekannt!
Bitte geben Sie naehere Informationen ein:
Wie lautet sein Name? Delphin
Mit welcher Frage kann man das Tier bestimmen? Ist es ein Saeugetier
Lautet die Antwort auf die Frage ja oder nein? ja

EXPERTEN-SYSTEM
(c) Olaf Hartwig '86

Anzahl der gespeicherten Tiere: 1
EXPERT-Frage: Kann das zu suchende Tier schwimmen?
ANWENDER-Antwort: ja
EXPERT-Frage: Ist es ein Saeugetier?
ANWENDER-Antwort: nein

Das gedachte Tier ist noch unbekannt!
Bitte geben Sie naehere Informationen ein:
Wie lautet sein Name? Hai
Mit welcher Frage kann man das Tier bestimmen? Ist es ein Allesfresser?
Lautet die Antwort auf die Frage ja oder nein? ja

Das Grundprinzip ist dabei, daß Sie an ein spezielles Tier denken und das System durch Fragen versucht, es zu bestimmen. Ist das Tier nicht bekannt, so werden die entsprechenden Fragen zu seiner Bestimmung gestellt. Mit jedem neuen Tier wächst der Suchbaum. Dabei wird jedem Tier ein spezieller Platz auf einem zugehörigen Baumast zugewiesen. An den Verzweigungen, den Knoten, stehen die Entscheidungsfragen, die Sie mit Ja oder Nein beantworten können. Anhand der Antworten findet EXPERT seinen jeweiligen Weg durch die Verästelungen, bis es entweder das gesuchte Tier findet oder feststellt, daß es noch unbekannt ist.
EXPERT erweitert mit jedem Dialogdurchgang seine Wissensbank. Es ist bereits bei diesem System möglich, daß es auf seinem Fachgebiet intelligenter als ein einzelner menschlicher Anwender ist. Dazu muß es nur genügend Informationen erhalten und sich zusätzlich z.B. mit mehreren Anwendern unterhalten. In seiner Datenbank enthält es damit das Wissen aller Dialogpartner, das über das eines einzelnen Anwenders hinausgehen kann.
Das EXPERT-System stellt mit seiner Suchbaumstruktur die Basis jedes modernen Experten-Systems dar. Es läßt sich ohne großen Aufwand als Shell verwenden. Dazu müssen nur einige Kommentare im Text sowie die Ausgangsfrage geändert werden. EXPERT eignet sich für alle Anwendungen, die auf Ja-/Nein-Entscheidungen basieren, also keine Wahrscheinlichkeiten verlangen. Einige mögliche Anwendungsgebiete sind einfache Fehleranalysen (z.B. ein Analysesystem für Fehler in Schaltkreisen oder Defekten an einem Auto), Diagnosen (z.B. Krankheiten), Wetteranalysen oder intelligente Lehranwendungen. Sie können EXPERT problemlos auf Ihre speziellen Anwendungen abstimmen. Es ist auch möglich, mehrere Suchbäume miteinander zu kombinieren. Ihnen steht hier ein großes Betätigungsfeld offen!

5.3.2 Eine Zusammenfassung

Wie bereits zu Anfang des Kapitels erwähnt wurde, sollte dieses Kapitel lediglich eine knappe Einführung in das komplexe Gebiet der Experten-Systeme geben. Es wurde dabei ein erster Ansatz zum Verständnis von Experten-Systemen unternommen. Sie können nun bereits aufbauend auf einer Database und einem Suchbaum ein Experten-System entwickeln, das auf seinem Fachgebiet gut Bescheid weiß.
Das auf einem Suchbaum basierende Experten-System ist für eindeutige Problemstellungen, wie für die erwähnten Gebiete, gut anwendbar, doch stellt man in der Praxis schnell fest, daß komplexe Probleme wesentlich kompliziertere Ansätze erfordern.
Nehmen wir einmal das Beispiel eines Psychiater-Experten-Systems oder eines medizinischen Diagnose-Systems. Sie müßten für eine annähernd korrekte Diagnose als Erweiterung des Suchbaum-Systems mit Wahrscheinlichkeiten operieren können.

> Wenn der Patient die spezifischen Symptome x,y und z zeigt, so besteht eine Wahrscheinlichkeit von p Prozent, daß er unter der Krankheit n leidet.

Fast alle komplexen Probleme sind nur über Wahrscheinlichkeiten zu lösen. Zum anderen muß die interne Wissensrepräsentation umfassender gelöst werden. Experten-Systeme im professionellen Einsatz benötigen eine sehr große Datenbank.
Um diese Datenmengen sinnvoll abrufen zu können, ist einerseits eine Trennung der Wissensbasis

vom Problemlösungsteil als auch der Benutzerschnittstelle erforderlich. Andererseits reicht ein einziger Suchbaum für die Daten nicht aus. Daher müssen oft mehrere Suchbäume von der Konzeption des vorangegangenen Experten-Systems mit unterschiedlichen Datenbanken verbunden werden.
Schauen wir uns die Konzeption eines Experten-Systems einmal im folgenden näher an.

5.4 Das Profil eines Experten-Systems

Bei all den vielfältigen und grundverschiedenen Anwendungsbereichen, in denen professionelle Experten-Systeme eingesetzt werden, sind doch zulässige Verallgemeinerungen über den Aufbau eines intelligenten Assistenten möglich.
Aufgrund der Database- und Experten-Suchbaumprogramme in diesem Kapitel sollte deutlich geworden sein, daß der entscheidende Schlüsselfaktor eines Systems das Wissen, repräsentiert durch die »Knowledge Base«, ist.
Das Wissen ist von zweierlei Art:

5.4.1 Die Wissenstypen eines Experten-Systems

Zum einen gibt es die zur Beherrschung eines speziellen Fachgebietes notwendigen Fakten. Damit ist das Wissen gemeint, das beispielsweise in Lehrbüchern oder Fachzeitschriften behandelt wird.
Die zweite Art von Wissen ist das sogenannte heuristische Wissen. Es beinhaltet Wissen über den Vorgang der Problemlösung zu einem Fachgebiet. Anders ausgedrückt wird hier dem Computer mitgeteilt, wie er bei der Problemlösung praktisch vorzugehen hat. Es handelt sich also um Erfahrungswissen, das jeder menschliche Spezialist in jahrelanger Arbeit erwirbt.
Das heuristische Wissen teilt dem Rechner mit, wie er das Faktenwissen genau auf das spezifische Problem, das das Experten-System zu lösen hat, anwenden muß.

5.4.2 Die Struktur der Experten-System-Datenbank

Um ein Problem mit einem hohen Sachverstand, vergleichbar dem eines menschlichen Experten, lösen zu können, muß das Experten-System in seiner Datenbank beide Arten von Wissen beinhalten.
In der Praxis verfährt man dabei zumeist so, daß die Datenbank in zwei Sub-Datenbanken unterteilt wird:

Die Wissensbasis

Sie enthält die allgemeingültigen Informationen über ein Fachgebiet. Sie beinhaltet also das heuristische Wissen zu einem Fachgebiet. Hier sind neben Praxiserfahrungen und Vorgehensweisen zur Problemlösung auch Meinungen und Vermutungen enthalten.

Die Datenbasis

Die Datenbank eines Experten-Systems gibt für die konkrete Situation eines Problemfalles die notwendigen und zugehörigen Regeln wieder.
Sie beinhaltet somit das spezifische Fachwissen, die Fakten zu dem Problemgebiet.

5.4.3 Anatomie eines Experten-Systems

Das Experten-Folgerungssystem

Die im vorherigen Abschnitt umrissene Wissensrepäsentation nimmt einen zentralen Stellenwert, die Schlüsselposition eines Experten-Systems ein. Aber aus dem Anatomiediagramm eines Systems im vorigen Abschnitt wird schon deutlich, daß Wissen allein noch kein Experten-Problem löst. Notwendig ist noch ein Steuerungsmechanismus, der aus den Regeln der gesamten Datenbank die richtige, d.h. die für den speziellen Problemfall zutreffende Regel auswählt und dann die entsprechenden Schlüsse zieht. Hier werden also Wissen und spezifische Problemdaten miteinander kombiniert.
Nützlich ist dabei die Anwendung einfacher Logik, auch bei komplexeren Problemen. Das hat den Vorteil, daß der Anwender selbst die Schlußfolgerungen des von ihm verwendeten Systems nachvollziehen und überprüfen kann.
Die modernsten Systeme setzen diese Technik konsequent ein. Vor allem bei Experten-Systemen in Kernkraftwerken wird die Entscheidung eines intelligenten Systems bei Unfällen oder auch Regulierungen innerhalb des Kraftwerks dem Anwender in einem graphischen Entscheidungsbaum dargestellt. Durch die anschauliche Darstellung und die Anwendung einfachster Regeln kann der Anwender in kritischen Fällen dem System mitteilen, ob er die vorgenommene Entscheidung unter Berücksichtigung seines menschlichen Erfahrungsschatzes akzeptiert.
Er kann auch bestimmte Faktoren manuell ändern und das Experten-System so direkt beeinflussen. Ein weiterer Vorteil der graphischen Darstellung von Entscheidungen besteht darin, daß der Anwender im Dialog mit dem System, wenn ihm die vorgenommenen Schlußfolgerungen nicht zusagen, die Regeln der Wissensbasis direkt ändern kann. Das System lernt also dazu und erweitert sein Wissen.

Die Systempflege

Ein professionelles Experten-System muß seinen Wissensschatz ständig erweitern. Das geschieht durch Feedback mit einem Experten. Eine spezielle Problemstellung wird jeweils sowohl einem menschlichen Spezialisten als auch einem Experten-System vorgelegt und anhand der Lösungsschritte werden die Qualität und Abweichung des Systems bestimmt.

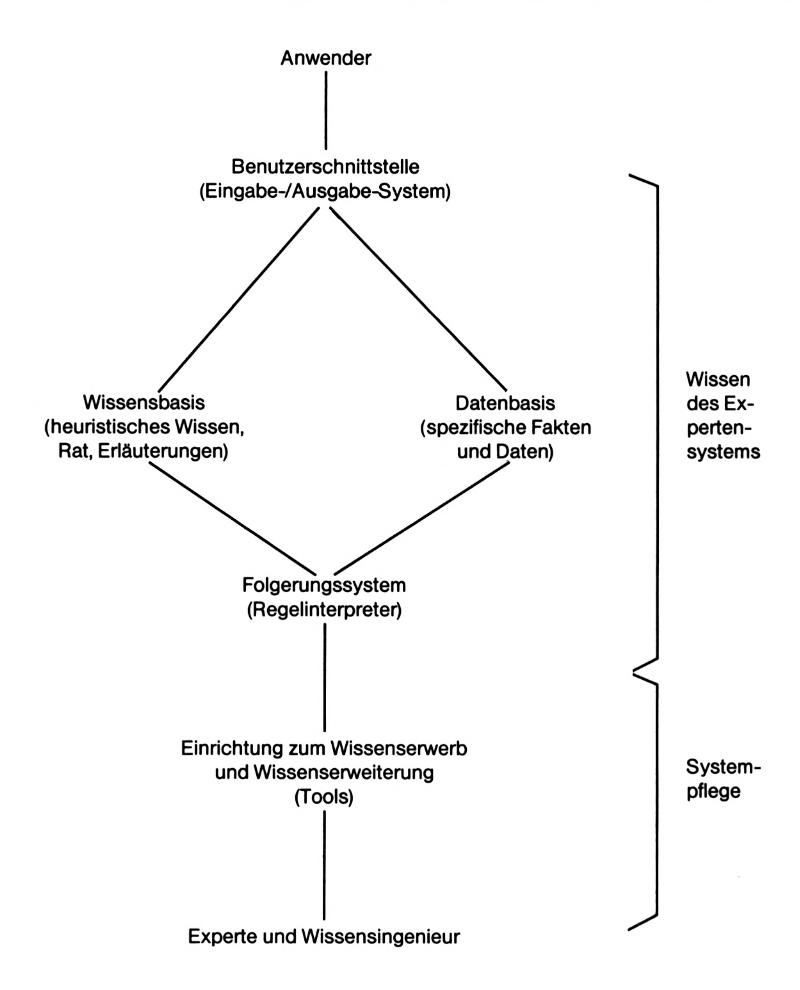


Bild 5.1

Die Regeln des intelligenten Assistenten werden dann entsprechend geändert. Die fähigsten Systeme sind auch in der Lage, nicht nur halbautomatisch, sondern autonom zu lernen und ihr Wissen zu erweitern bzw. aufgrund neuer Fakten alte Regeln zu überprüfen.

Abriß der Anatomie eines Experten-Systems

Die entscheidende Frage für die Künstliche Intelligenz eines Experten-Systems ist die Frage der Wissensdarstellung. Wie wird das Wissen im Computer repräsentiert?
Hier hat sich die Teilung des Wissens in heuristisches Wissen, also Erfahrungswissen, und in Faktenwissen durchgesetzt. Die Datenbank ist vom Steuerungsteil des Systems getrennt, vor allem wegen der besseren Softwarepflege, also der Erweiterung und Änderung des Wissens wegen.

An zweiter Stelle steht das Problem der Wissensnutzung. Wie kann das vorhandene Wissen zur Problemlösung herangezogen werden? Dieses Problem wird vom Folgerungs-System bewältigt.
Der dritte Punkt ist der Wissenserwerb. Wie kann das für die Problemlösung notwendige Wissen automatisch oder, wenn dies nicht möglich ist, halbautomatisch erworben werden? Der Erwerb von Wissen, autonomes Lernen, ist ein altes Problem der KI. Dieses Problem ist auch heute noch nicht gelöst. Derzeit ist die Problematik des Wissenserwerbs noch der entscheidende Engpaß, nicht nur bei Experten-Systemen, sondern in der Künstlichen Intelligenz allgemein.
Die Datenbank eines Experten-Systems wird von einem Wissensingenieur in Zusammenarbeit mit einem Experten in mühevoller Kleinarbeit erworben.
Nur wenn ein System mit einem recht umfangreichen Grundwissen ausgestattet ist, kann es später in der Lage sein, die Datenbank selbständig zu ändern und zu erweitern. Aber auch diese automatischen oder halbautomatischen Lernvorgänge sind noch lange nicht optimal gelöst.
Wie Sie festgestellt haben werden, ist die Problematik der Experten-Systeme sehr komplex, besonders Erstellung, Wissensrepräsentation, Folgerungs-System, Systempflege als auch, ganz entscheidend, Lernfähigkeit.

5.5 Das DEX.C3-Experten-System

Die GMD, die Großforschungsstelle für Mathematik und Datenverarbeitung in Deutschland, hat das professionelle Experten-System DEX.C3 entwickelt. Das für den Automobilhersteller FORD (Europa) konzipierte System dient der Fehlerdiagnose im automatischen Getriebe von FORD-Motoren.
Die Wissensbasis des Experten-Systems besteht dank des sehr eingeschränkten Wirkungsbereichs aus insgesamt 140 Regeln.
Auf der folgenden Seite ist ein Bildschirmausdruck dargestellt, der wie die Beschreibung des DEX.C3-Systems mit freundlicher Genehmigung der GMD abgedruckt wird:
Bild 5.2 zeigt, wie sich das System dem Benutzer auf dem hochauflösenden Rasterdisplay präsentiert. Der Bildschirm ist in zwei große Regionen aufgeteilt. Das obere Fenster beinhaltet eine Darstellung des Getriebes mit allen wichtigen Erklärungen. Darunter befindet sich das Dialogfenster, über das der Benutzer mittels einer Maus kommuniziert.
Rechts daneben ist das TRACE-Fenster dargestellt. In ihm »denkt das System laut«, d.h. der Benutzer kann die einzelnen Folgerungsabschnitte des Experten-Systems nachvollziehen.
Das gewählte Wissensgebiet eines C3-Getriebes scheint zwar auf den ersten Blick etwas ungewöhnlich, die Wahl des Gebiets hat aber die folgenden Hintergründe:

- Das Expertenwissen über die Fehlerdiagnose im C3-Getriebe ist nicht bei jedem Mechaniker vorhanden.
- Es handelt sich um ein überschaubares Feld, so daß die Entwicklung in verhältnismäßig kurzer Zeit abgeschlossen werden konnte.
- Das automatische Getriebe C3 ist ein im wesentlichen abgeschlossenes System, das mechani-

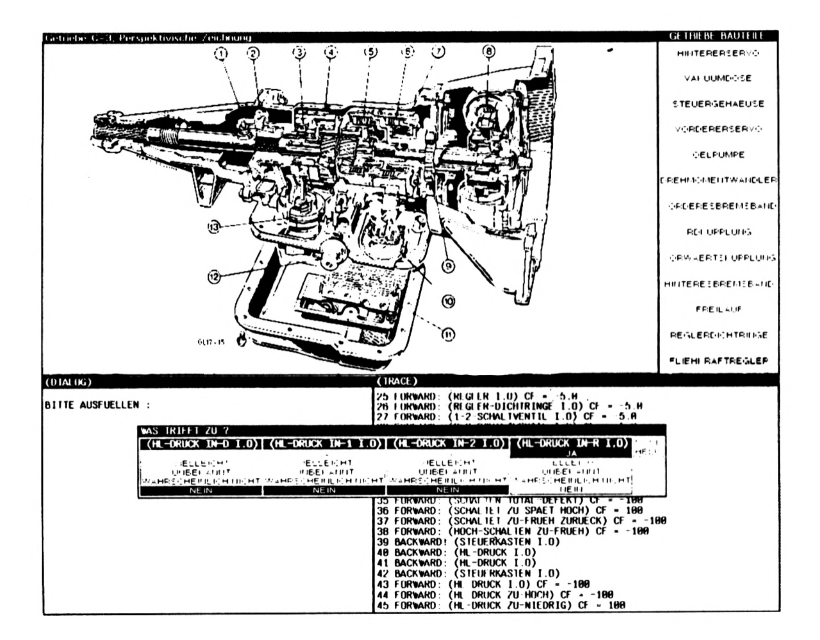


Bild 5.2

sche und hydraulische Komponenten in sich vereint. Die Abgeschlossenheit vereinfacht die Diagnose wesentlich.

- Es kommt vor, daß Fehler im automatischen Getriebe nicht richtig erkannt werden, wodurch unnötig hohe Kosten entstehen. Diese können durch den Einsatz dieses Systems minimiert werden.

5.5.1 Die DEX.C3-Wissensbasis

Die Wissensbasis des Systems enthält das relevante Erfahrungswissen des Kundendienstfachmannes. Das Wissen ist im wesentlichen durch die potentiellen Fehlerfälle bestimmt, auf die sich die Diagnosen beziehen.

Die Hauptproblematik der Diagnosestellung liegt darin, daß die wirklichen Ursachen eines Defekts oft nicht erkannt werden. Das hat zur Folge, daß unnötige Reparaturen durchgeführt werden, welche den Defekt nicht oder nur für einen kurzen Zeitraum beheben. Teilweise können durch derartige Reparaturen sogar kostspielige Folgefehler auftreten.

Das System DEX.C3 teilt die Diagnosen in drei Kategorien ein. Es sind

- Kraftübertragung (11 Diagnosen),
- Hydraulik (17 Diagnosen) und
- Verschiedenes (5 Diagnosen).

Die Diagnose »Kraftübertragung« ist noch einmal in

- kraftschlüssig (6 Diagnosen) und
- formschlüssig (5 Diagnosen)

unterteilt.
Die Diagnose »Hydraulik« ist ebenfalls in weitere Unterkategorien unterteilt:

- Ölversorgung, Ölstand (5 Diagnosen),
- Stellhydraulik (5 Diagnosen) und
- Schalthydraulik (7 Diagnosen).

Das Diagnosewissen ist in 140 Regeln abgefaßt. Sie lassen sich in

- 37 Steuerregeln,
- 24 Konsistenzregeln und
- 79 Diagnoseregeln

einordnen.
Die Steuerregeln legen das Frageverhalten des Systems fest, bestimmen die Reihenfolge der zu überprüfenden Hypothesen und unterdrücken unsinnige Tests. Sie dienen damit zur Kontrolle des Systemverhaltens.
Konsistenzregeln beinhalten logische Zusammenhänge allgemeiner Art.
Diagnoseregeln sind die Regeln, die zur Herleitung von End- und Zwischendiagnosen aus den Symptomen benötigt werden.
Alle Regeln sind einheitlich strukturiert. Eine Regel besteht immer aus einem Bedingungs- und einem Aktionsteil.
Bei der Abarbeitung der Regeln werden zwei grundsätzlich verschiedene Strategien miteinander kombiniert:

Forward Chaining

Bei dieser Strategie geht das System von bekannten Fakten aus. Es sucht dann eine Regel, deren Bedingungsteil aufgrund der Fakten erfüllt ist.
So werden neue Fakten abgeleitet, die sich aus dem Aktionsteil einer Regel ergeben.

Backward Chaining

Hierbei wird von einer Hypothese ausgegangen und versucht, sie zu verifizieren. Dabei muß eine Regel gefunden werden, die die Hypothese im Aktionsteil enthält und deren Bedingungsteil erfüllt ist.

Die Kombination der beiden Verfahren läuft nach folgendem Konzept ab: zunächst werden »vorwärts« Übersichtsfragen gestellt, die zur Verdachtsgenerierung benutzt werden.
Die jeweils aussichtsreichste Hypothese wird daraufhin rückwärts überprüft. Beim »Backward Chaining«-Verfahren werden an den Benutzer weitere Fragen gestellt. Diese dienen dazu, den Verdacht zu erhärten oder zu widerlegen. Wenn eine Hypothese auf diese Weise abgearbeitet

wurde, werden die inzwischen vom Benutzer erhaltenen Informationen wieder nach dem »Forward Chaining« zur erneuten Verdachtsgenerierung benutzt.
Dieser Zyklus von Verdachtsgenerierung und Verdachtsüberprüfung spiegelt das Vorgehen des menschlichen Experten bei der Fehlersuche wider.

5.5.2 Wahrscheinlichkeiten

DEX.C3 kann auch mit unsicherem Wissen umgehen. Die Informationen, mit denen ein Experte arbeiten muß, sind häufig ungenau und vage. Die Behandlung von »Unsicherheit« wird über Wahrscheinlichkeiten gelöst.
Das bedeutet, daß der Benutzer Fragen nicht nur mit

»JA«

oder

»NEIN«

beantworten kann, sondern auch

»VIELLEICHT«
»UNBEKANNT«

oder

»WAHRSCHEINLICH«

als Antwort geben kann. Die einzelnen Wahrscheinlichkeiten werden vom System auf plausible Weise verarbeitet.

5.5.3 Die Erklärungskomponente

Das Verhalten von Experten-Systemen muß für den Benutzer immer transparent und kontrollierbar bleiben. Deshalb spielt die Erklärungskomponente bei jedem der Systeme eine wesentliche Rolle. In DEX.C3 gibt es hierbei die folgenden Möglichkeiten:

1. Der Benutzer kann sich erklären lassen, warum das System eine bestimmte Frage stellt. Es werden dann die Regeln aufgelistet, aus denen zusammen mit dem erfragten Faktum die untersuchte Hypothese abzuleiten ist.
2. Der Wissensstand des Systems kann zu jeder Zeit erfragt werden.
3. Alle Regeln, die ein bestimmtes Faktum enthalten, lassen sich auflisten.
4. Während des gesamten Dialogs wird das Getriebe in perspektivischer Darstellung gezeigt. Bauteile, die gerade untersucht werden und als hinreichend verdächtig erscheinen, blinken dabei auf. Auf diese Weise kann der Benutzer stets den Fortgang der Untersuchung beobachten.
5. Der TRACE-Modus spiegelt zeilenweise die Ableitung von Fakten wider, wie sie sich, durch die Dialogeingaben gesteuert, durch Ausführen der jeweiligen Regeln ergeben.
6. Der Benutzer kann ferner erfragen, warum ein System ein Faktum abgeleitet hat.

An Bild 5.2 wird deutlich, daß die Anwendung von DEX.C3 sehr bedienungsfreundlich gestaltet ist. Die Benutzeroberfläche ist weitgehend durch die interaktive Auswahl von Alternativen mit Hilfe der Menütechnik realisiert. Der Bildschirm ist dabei in einzelne Windows aufgeteilt, in denen Fragen, Erklärungen, Ergebnisse und graphische Darstellungen des Systems erscheinen.
In der Praxiserprobung hat sich gezeigt, daß das System nach der Beurteilung menschlicher C3-Experten die Fehlerdiagnose wie ein Mensch vollzieht. Es werden die richtigen Fragen zur richtigen Zeit gestellt und die korrekten Diagnosen generiert.

6 Wissensrepräsentation auf dem CPC

6.1 Anwendungen der Wissensrepräsentation

Im vorherigen Kapitel wurde dargestellt, welchen zentralen Stellenwert die Wissensrepräsentation (»Knowledge-Representation«) in der Künstlichen Intelligenz einnimmt.
In diesem Kapitel mache ich Sie nach der im Experten-System EXPERT realisierten Suchbaumstruktur mit einer weiteren effizienten Wissensrepräsentation vertraut. Das Programm stellt eine Version des SIR, des »Semantic Information Retrieval« dar. Das mit Künstlicher Intelligenz ausgestattete Programm wurde in seinem Grundkonzept ursprünglich vom KI-Experten Bertram Raphael entwickelt.
Es ist eine der bekanntesten Versionen der intelligenten »Q/A-Programme«. Q/A-Programme sind »Question-answering«-Programme. Der Anwender kann Informationen in ein Q/A-System eingeben und zu jeder Zeit Fragen stellen, die auf intelligente Weise beantwortet werden.
Die Daten können in natürlicher Sprache eingegeben werden. Raphaels Programm versteht Englisch, das in diesem Buch vorgestellte SIR-Programm kann nach einem ähnlichen Prinzip deutsche Sätze verarbeiten. Wenn Fragen zu den gespeicherten Daten gestellt werden, ist das SIR-Programm in der Lage, mit einem sinnvollen Text in Deutsch zu antworten.
Die ganz besondere Eigenschaft des CPC-SIR-Programms liegt allerdings darin, daß das Programm automatisch intelligente Verknüpfungen der eingegebenen Daten vornimmt. Logische Zusammenhänge zwischen den Daten werden erkannt und intern entsprechend repräsentiert. Die Schlußfolgerungen von SIR werden auf spezielle Fragen dem Anwender mitgeteilt.
Das SIR-Programm kann beispielsweise den folgenden Dialog führen:

ANWENDER:	Peter ist ein Stanford Student
SIR:	Ich verstehe.
ANWENDER:	Jeder Stanford Student ist intelligent
SIR:	Ich verstehe.

ANWENDER: Ist Peter intelligent?
SIR: Ja.

.
.
.

Es können nicht nur zwei Daten wie im Beispiel, sondern Hunderte von Fakten von SIR miteinander verknüpft werden.
Bei dem Dialog handelt es sich um einen »Syllogismus«. Einer der bekanntesten Syllogismen stammt aus der Zeit von Aristoteles und lautet folgendermaßen:

Sokrates ist ein Mensch.
Jeder Mensch ist sterblich.
Daher ist auch Sokrates sterblich.

Die ersten beiden Zeilen stellen die Prämissen dar, die dritte Zeile ist die Schlußfolgerung. Entscheidend ist hierbei, daß durch die Schlußfolgerung keine neuen Daten geschaffen werden, sondern vielmehr alte Informationen logisch miteinander verknüpft werden.
Dies entspricht dem Vorgang des »deduktiven Denkens«. Dabei werden Beziehungen zwischen unterschiedlichen Datenklassen hergestellt.
Die Logik in unserem Beispiel ist einfach:

A ist B

und

B ist C,

daraus folgt

A ist C.

Dabei entspricht

A der Aussage »Sokrates ist ein Mensch.« und
B der Aussage »Jeder Mensch ist sterblich.« und
C der Folgerung »Sokrates ist sterblich.«

Das nun folgende SIR-Programm kann noch mehr, als Syllogismen zu entdecken. Aber realisieren wir vorerst den Syllogismen-Teil:

6.2 Das SIR-Programm

Der Programmkopf des »Semantic Information Retrieval«-Programms ruft die vier Module »Initialisierung«, »Titel«, »Eingabe« und »Processing« auf. Er lautet:

```
10 REM Kuenstliche Intelligenz auf dem CPC
20 REM Semantic Information Retrieval - SIR
25 :
30   BORDER 9,9
33   WIDTH 60
35   MODE 1
40   PAPER 0
45   PEN 1
50 :
100     GOSUB 1000
110        REM Initialisation
200     GOSUB 2000
210        REM Titel
250     GOSUB 2400
260        REM Eingabe
300     GOSUB 3000
310        REM Processing
400 GOTO 250
```

Die Initialisierung

In diesem Programmteil wird die Repräsentation der Wissensstruktur von SIR initialisiert. Dabei erfolgt die Festlegung eines Textfeldes mit 21 * 21 Elementen. Anschließend wird die oberste Zeile des Feldes mit dem Zeichen »#« markiert.
Das Modul lautet:

```
1000 REM Initialisation
1005   DIM d$(21,21)
1010    FOR i=0 TO 20
1020      d$(0,i)="#"
1030    NEXT i
1500 RETURN
```

Der Titel

Hier werden das Graphikdesign des Programms festgelegt, der Titel ausgegeben und ein Textfenster gesetzt.
Das sieht wie folgt aus:

```
2000 REM Titel
2005  CLS
2010    PRINT "Semantic Information Retrieval - SIR"
2015    PRINT
2016    ORIGIN 0,380:DRAW 660,0
2018    ORIGIN 300,350:DRAW 330,0
2020    PRINT TAB(20)"(c) Olaf Hartwig '85"
2050      WINDOW 3,40,5,40
2060      PAPER 3
2070  CLS
2100 RETURN
```

Die Texteingabe

Ihre in natürlicher Sprache abgefaßten Informationen und Anweisungen werden in die Variable »ein$« übernommen. Geben Sie hier »s« oder »stop« ein, so wird der Dialog beendet:

```
2400 REM Texteingabe
2420   INPUT"SIR__>";ein$
2500   IF ein$="s" OR ein$="stop" THEN PRINT:STOP
2700 RETURN
```

6.2.1 Der SIR-Verarbeitungsteil

Dieser Teil stellt das Kernmodul des Programms dar. Er gliedert sich in drei große Untermodule, die von einem Modul-Supervisor aufgerufen werden.
Der Modulkopf lautet:

```
3000 REM Verarbeitung
3005 :
3010  REM  Parsing nach Keywoertern
3015    hilf$=LEFT$(ein$,4)
3020    IF hilf$="ist " OR hilf$="Ist " THEN GOSUB 10000
3030    IF hilf$="Info" OR hilf$="info" THEN GOSUB 20000
3040    FOR i=1 TO LEN(ein$)
3050      IF MID$(ein$,i,4)="ist " THEN GOSUB 30000
3060    NEXT i
3070 RETURN
```

Der Kopfteil untersucht den zuvor im Eingabeteil eingegebenen Text nach Schlüsselwörtern. Dazu muß gesagt werden, daß bei der Eingabe zwischen drei verschiedenen Satztypen unterschieden wird:

1. Fragesatz

   Ein Beispiel für einen derartigen Satz ist unsere Frage

   »Ist Sokrates sterblich?«

   Ein Fragesatz wird daran erkannt, daß er mit dem Hilfsverb »ist« beginnt. Ist eine Frage als solche erkannt worden, so wird das die Fragesätze behandelnde Untermodul in den Zeilen 10000 ff. aufgerufen.

2. Information

   Ein Beispiel für eine Informationsfrage ist die Anweisung

   »Info Sokrates.«

   Dabei werden alle Sokrates betreffenden Daten in einem speziellen Infofenster ausgegeben. Dieses Untermodul befindet sich in den Zeilen 20000 ff.

3. Aussagesatz

   Ein Aussagesatz liefert SIR die entsprechenden Informationen. Ein Beispiel dafür ist

   »Sokrates ist ein Mensch.«

   Der Aussagesatz wird an der Zuweisung A ist B, also in diesem Fall »SOKRATES« ist »EIN MENSCH«, erkannt. Das den Aussagesatz aufbereitende Modul befindet sich in den Zeilen 30000 ff.

6.2.2 Die SIR-Wissensrepräsentation

Nach der Initialisierung ist ein Datenfeld vorhanden, von dem ein Ausschnitt in Bild 6.1 dargestellt ist.

	0	1	2	3	4
0	Marcus	Mensch	Bewohner Pompeius	Sterblicher	Jahr
1	'#'	'#'	'#'	'#'	'#'
2					
3					
4					
5					

Bild 6.1

Wenn das Feld mit einzelnen Daten belegt ist, erhalten wir eine Datenstruktur, wie sie in Bild 6.2 dargestellt ist. Die erste Zeile des Arrays enthält die einzelnen Oberbegriffe. Bei der Dateneingabe wurden dabei Informationen zu den folgenden Oberbegriffen gegeben:

- Marcus
- Mensch
- Bewohner Pompejis
- Sterblicher
- Jahr

Die eingegebenen Informationssätze lauten in unserem Beispiel wie folgt:
- Marcus ist ein Mensch.
- Marcus ist ein Bewohner Pompejis.
- Ein Mensch ist sterblich.
- Jeder Bewohner Pompejis ist beim Vulkanausbruch gestorben.
- Jeder Bewohner Pompejis ist ein Mensch.
- Das Jahr ist 1985.

	0	1	2	3	4
0	Marcus	Mensch	Bewohner Pompeius	Sterblicher	Jahr
1	Mensch	sterblich	beim Vulkan-ausbruch gestorben	'#'	1985
2	Bewohner Pompeius	'#'	Mensch		'#'
3	'#'		'#'		
4					

Bild 6.2

Das Aussagesatz-Modul

Das Modul schlüsselt die eingegebenen Aussagesätze so auf, daß sie in eine für das Datenfeld akzeptable Form gebracht werden. Der Prozeß läuft so ab, daß festgestellt wird, wo sich das trennende Hilfsverb »ist« im Aussagesatz befindet.
Der Satz

Sokrates IST ein Mensch

wird aufgeteilt in das Objekt

Sokrates

und in das dazugehörige Element

ein Mensch.

Zunächst werden die bereits gespeicherten Objekte untersucht, ob eines von ihnen identisch mit »Sokrates« ist. Ist das der Fall, so wird die Anzahl der Elemente um eins erhöht und das neue Element »ein Mensch« eingetragen. Daran schließt sich der Marker »#« an, der anzeigt, daß das Datenfeld nach diesem Element beendet ist. Der alte Marker wurde zuvor durch den Inhalt des neuen Elements überschrieben.
Wenn das Objekt »Sokrates« nicht gefunden wurde, wird es neu eingerichtet und das Element »ein Mensch« als erstes gespeichert.
Das entsprechende Programm-Modul sieht folgendermaßen aus:

```
30000 REM Zuweisung : A ist B
30020  a$=LEFT$(ein$,i-2)
30040  b$=MID$(ein$,i+4)
30100 :
30500  FOR x=0 TO 20
30510    IF a$=d$(0,x) THEN 30600
30520  NEXT x
30525 :
30530  FOR x=0 TO 20
30540    IF d$(0,x)="#" THEN 30560
30550  NEXT x
30555 :
30560   d$(0,x)=a$:d$(0,x+1)="#"
30570   d$(1,x)=b$:d$(2,x)="#"
30575  PRINT"OK!"
30580 RETURN
30590 :
30600  FOR y=1 TO 9
30610    IF d$(y,x)="#" THEN d$(y,x)=b$:d$(y+1,x)="#":GOTO 30700
30650  NEXT y
30700  PRINT"OK!"
30710 RETURN
```

Das Informations-Modul

Dieses Modul gibt alle zu einem Objekt gespeicherten Elemente aus. Dazu wird ein spezielles Informationsfenster definiert. Das sieht so aus:

```
20000 REM Objekt-Information
20003  yp=VPOS(#0)
20005   WINDOW 30,40,6,25
20007    PAPER 9:PEN 0:CLS
20008     IF LEN(ein$)=4 THEN 20250
20010    ein$=RIGHT$(ein$,LEN(ein$)-5)
20011 :
```

```
20012     PRINT" INFOLISTE"
20014     PRINT"-----------";
20015       t$="Ich    habekeine DATENbezueglich:"
20020     FOR x=0 TO 20
20030       IF d$(0,x)<>ein$ THEN NEXT x:PRINT t$:PRINT ein$:
            GOTO 20250
20035 :
20040     PRINT ein$;">":PRINT
20045 :
20050     FOR y=1 TO 20
20060       IF d$(y,x)<>"#" THEN PRINT d$(y,x): NEXT y
20090 :
20100     PRINT
20200     PRINT"Datenende."
20250    WINDOW SWAP 0,2
20260    WINDOW 3,29,5,25
20280    PAPER 3
20285    PEN 1
20290   LOCATE 1,yp
20300  RETURN
```

Das Fragesatz-Modul

Eine an SIR gestellte Frage hat das Format

IST A (Artikel) B ?

Zunächst wird das Hilfsverb »ist« vom eingegebenen Satz abgeschnitten. Anschließend wird der Restsatz nach dem Artikel

ein_
eine
der_
die_

abgesucht.
Das Prinzip ist dabei das des »scanning«, das Sie bereits beim ELIZA-Programm kennengelernt haben.
Der so analysierte Satz wird in das Objekt und ein dazugehöriges Element unterteilt. Anschließend wird das gesamte Datenfeld nach dem Objekt und dem Element abgesucht, wobei nach dem Prinzip des bereits erörterten »deduktiven Denkens« logische Verknüpfungen vorgenommen werden.
Das Modul lautet folgendermaßen:

```
10000 REM Keywort "Ist...?" : Frage
10010    ein$=RIGHT$(ein$,LEN(ein$)-4)
10020       REM ersten vier Buchstaben abschneiden
10030 :
10100    REM Zuweisungssatz teilen
10120     FOR x=1 TO LEN(ein$)
10130      IF MID$(ein$,x,4)="ein " THEN 11000
10140      IF MID$(ein$,x,5)="eine " THEN 11000
10150      IF MID$(ein$,x,4)="der " THEN 11000
```

```
10160       IF MID$(ein$,x,4)="die " THEN 11000
10170     NEXT x
10200 :
10300     FOR x=1 TO LEN(ein$)
10310       IF MID$(ein$,x,1)=" " THEN 11000
10330     NEXT x
10400     PRINT"Unidentifizierbare Frage!"
10410 RETURN
11000 a$=LEFT$(ein$,x-1)
11010 b$=MID$(ein$,x):b$=MID$(b$,2)
12000     FOR i=0 TO 20
12020       IF d$(0,i)=a$ THEN 15000
12030     NEXT i
12050   PRINT"Objekt ist nicht bekannt."
12060 RETURN
14000 :
15000   FOR y=0 TO 20
15010     FOR x=0 TO 20
15020       IF d$(y,x)=b$ THEN 15100
15030     NEXT x
15040   NEXT y
15050 GOTO 15500
15070 :
15100   FOR y=1 TO 20
15110     IF d$(y,i)=d$(0,x) THEN PRINT"JA!":RETURN
15120     IF d$(y,i)=b$ THEN PRINT "JA, im selben Objekt!":RETURN
15150   NEXT y
15500   PRINT"Nicht bekannt!"
15510 RETURN
```

Damit ist das SIR-Programm vollständig. Ein beispielhafter Demonstrationslauf ist im folgenden dargestellt:

Semantic Information Retrieval - SIR (Olaf Hartwig '86)

SIR..> ? Sokrates ist ein Mensch
OK!
SIR..> Info Sokrates

INFOLISTE
Sokrates>
ein Mensch
Datenende.

SIR..> ? ein Mensch ist sterblich
SIR..> ? Sokrates ist Wissenschaftler
OK!
SIR..> ? ist Sokrates sterblich
JA!

.

.

Unten sehen Sie das komplette Listing des SIR-Programms mit allen Modulen in der korrekten Reihenfolge:

```
10 REM Kuenstliche Intelligenz auf dem CPC
20 REM Semantic Information Retrieval - SIR
25 :
30   BORDER 9,9
33   WIDTH 60
35   MODE 1
40   PAPER 0
45   PEN 1
50 :
100    GOSUB 1000
110       REM Initialisation
200    GOSUB 2000
210       REM Titel
250    GOSUB 2400
260       REM Eingabe
300    GOSUB 3000
310       REM Processing
400 GOTO 250
500 :
1000 REM Initialisation
1005   DIM d$(21,21)
1010    FOR i=0 TO 20
1020       d$(0,i)="#"
1030    NEXT i
1500 RETURN
1600 :
2000 REM Titel
2005  CLS
2010    PRINT "Semantic Information Retrieval - SIR"
2015    PRINT
2016    ORIGIN 0,380:DRAW 660,0
2018    ORIGIN 300,350:DRAW 330,0
2020    PRINT TAB(20)"(c) Olaf Hartwig '85"
2050      WINDOW 3,40,5,40
2060      PAPER 3
2070  CLS
2100 RETURN
2200 :
2400 REM Texteingabe
2420   INPUT"SIR__>";ein$
2500   IF ein$="s" OR ein$="stop" THEN PRINT:STOP
2700 RETURN
2900 :
3000 REM Verarbeitung
3005 :
3010  REM  Parsing nach Keywoertern
3015    hilf$=LEFT$(ein$,4)
3020    IF hilf$="ist " OR hilf$="Ist " THEN GOSUB 10000
3030    IF hilf$="Info" OR hilf$="info" THEN GOSUB 20000
3040    FOR i=1 TO LEN(ein$)
3050      IF MID$(ein$,i,4)="ist " THEN GOSUB 30000
3060    NEXT i
```

```
3070 RETURN
4000 :
10000 REM Keywort "Ist...?" : Frage
10010   ein$=RIGHT$(ein$,LEN(ein$)-4)
10020      REM ersten vier Buchstaben abschneiden
10030 :
10100   REM Zuweisungssatz teilen
10120    FOR x=1 TO LEN(ein$)
10130     IF MID$(ein$,x,4)="ein " THEN 11000
10140     IF MID$(ein$,x,5)="eine " THEN 11000
10150     IF MID$(ein$,x,4)="der " THEN 11000
10160     IF MID$(ein$,x,4)="die " THEN 11000
10170   NEXT x
10200 :
10300   FOR x=1 TO LEN(ein$)
10310     IF MID$(ein$,x,1)=" " THEN 11000
10330   NEXT x
10400   PRINT"Unidentifizierbare Frage!"
10410 RETURN
11000 a$=LEFT$(ein$,x-1)
11010 b$=MID$(ein$,x):b$=MID$(b$,2)
12000   FOR i=0 TO 20
12020     IF d$(0,i)=a$ THEN 15000
12030   NEXT i
12050  PRINT"Objekt ist nicht bekannt."
12060 RETURN
14000 :
15000  FOR y=0 TO 20
15010   FOR x=0 TO 20
15020     IF d$(y,x)=b$ THEN 15100
15030   NEXT x
15040  NEXT y
15050 GOTO 15500
15070 :
15100  FOR y=1 TO 20
15110    IF d$(y,i)=d$(0,x) THEN PRINT"JA!":RETURN
15120    IF d$(y,i)=b$ THEN PRINT "JA, im selben Objekt!":RETURN
15150  NEXT y
15500  PRINT"Nicht bekannt!"
15510 RETURN
19800 :
20000 REM Objekt-Information
20003  yp=VPOS(#0)
20005   WINDOW 30,40,6,25
20007    PAPER 9:PEN 0:CLS
20008     IF LEN(ein$)=4 THEN 20250
20010    ein$=RIGHT$(ein$,LEN(ein$)-5)
20011 :
20012    PRINT" INFOLISTE"
20014    PRINT"-----------";
20015      t$="Ich    habekeine DATENbezueglich:"
20020    FOR x=0 TO 20
20030      IF d$(0,x)<>ein$ THEN NEXT x:PRINT t$:PRINT ein$:
           GOTO 20250
20035 :
20040    PRINT ein$;">":PRINT
```

```
20045 :
20050     FOR y=1 TO 20
20060       IF d$(y,x)<>"#" THEN PRINT d$(y,x): NEXT y
20090 :
20100     PRINT
20200     PRINT"Datenende."
20250    WINDOW SWAP 0,2
20260    WINDOW 3,29,5,25
20280    PAPER 3
20285    PEN 1
20290   LOCATE 1,yp
20300 RETURN
29980 :
30000 REM Zuweisung : A ist B
30020  a$=LEFT$(ein$,i-2)
30040  b$=MID$(ein$,i+4)
30100 :
30500  FOR x=0 TO 20
30510    IF a$=d$(0,x) THEN 30600
30520  NEXT x
30525 :
30530  FOR x=0 TO 20
30540    IF d$(0,x)="#" THEN 30560
30550  NEXT x
30555 :
30560   d$(0,x)=a$:d$(0,x+1)="#"
30570   d$(1,x)=b$:d$(2,x)="#"
30575  PRINT"OK!"
30580 RETURN
30590 :
30600  FOR y=1 TO 9
30610    IF d$(y,x)="#" THEN d$(y,x)=b$:d$(y+1,x)="#":GOTO 30700
30650  NEXT y
30700  PRINT"OK!"
30710 RETURN
```

6.2.3 Modifikationen des Programms

Das »Semantic Information Retrieval« ist ein gutes Beispiel dafür, wie man Wissen auf dem Computer repräsentieren kann.

Es gibt aber noch Verbesserungsmöglichkeiten. So ist beispielsweise das Eingabeformat auf einige wenige, deutschsprachige Satzkonstruktionen beschränkt. So können Fragesätze nicht nur mit »Ist...?« beginnen, sondern auch mit anderen Hilfsverben bzw. anderen Formen des Hilfsverbs »sein«.

Das sollten Sie jedoch nach dem Kapitel über das Verarbeiten natürlicher Sprache mit etwas Programmieraufwand beseitigen können.

Ferner lassen sich die logischen Fähigkeiten des Programms noch erweitern. Hier haben Sie für eigene Experimente ein weites und interessantes Betätigungsfeld.

7 Das Verzeichnis der KI-Programme für den CPC

An dieser Stelle gebe ich Ihnen eine Übersicht über alle in dem Buch vorgestellten Programme der Künstlichen Intelligenz auf dem CPC.

Verarbeiten natürlicher Sprache:

DOC

Programmtyp: Pseudo-Dialog

PARSING VON SÄTZEN

Programmtyp: Parsing von Texten in Sätze

PARSING VON WORTEN

Programmtyp: Parsing von Sätzen in Worte

PARSING VON TEXTEN

Programmtyp: Gesamtparsing von Texten

PARSING MIT AUTOMATISCHEM LERNEN

Programmtyp: Parsing von Texten mit kombiniertem Lernen und Erweiterung des Parsingvokabulars

BASIC RAC

Programmtyp: Umwandlung von Eingaben in natürlicher Sprache und Ansätze zu sinnvollen Antworten am Beispiel von RACTER

MDBABYVERSION 1

Programmtyp: Generierung intelligenter Antworten

MDBABYVERSION 2
Programmtyp: Modifizierte Generierung intelligenter natürlichsprachiger Antworten
ELIZA PSYCHIATER
Programmtyp: ELIZA mit Scriptmodul eines Psychiaters
ELIZAVORGESETZTER
Programmtyp: ELIZA mit Scriptmodul eines Vorgesetzten

Computer-Kreativität:
SCRUDU PROTO
Programmtyp: Automatische Textgenerierung, Gedichtsgenerierung ohne Beachtung des grammatischen Formats
SCRUDU
Programmtyp: Automatische Gedichtsgenerierung mit grammatischer Umformung

Robotics - Der Micro-Mouse-Wettbewerb:
MMC-BRAIN
Programmtyp: Intelligente Micro-Mouse-Simulation mit einfachem »Gedächtnis« durch Markierung des zurückgelegten Weges der Maus
MMC-RND
Programmtyp: Intelligente Micro-Mouse-Simulation durch kontrollierte Zufallswahl
PLEDGE
Programmtyp: Intelligente Micro-Mouse-Simulation mittels Befolgen des Pledge-Algorithmus
MAXI-PLEDGE
Programmtyp: Intelligente Micro-Mouse-Simulation auf einem großen Labyrinth

Experten-Systeme:
DATABASE VERSION 1
Programmtyp: Demo-Database zum Aufzeigen der Wissensrepräsentation von Daten
DATABASE VERSION 2
Programmtyp: Optimierte und flexiblere Demo-Database
EXEC-EXPERTENSYSTEM
Programmtyp: »Starres Experten-System«, alles Wissen ist fest vorhanden, es besteht noch keine autonome Lernfähigkeit
EXPERT
Programmtyp: Intelligenter Suchbaum mit der Grundstruktur professioneller Experten-Systeme. Er beinhaltet u.a. die Fähigkeit zum eigenständigen Erkennen und Korrigieren von fehlerhaften Eingaben kombiniert mit Lernvermögen

Wissensrepräsentation:
SIR
Programmtyp: Semantic Information Retrieval, intelligentes »Question-Answer«-Programm mit der Fähigkeit zu logischen Schlußfolgerungen

8 Die KI-Chronologie

In diesem Kapitel erhalten Sie eine Zusammenstellung aller wichtigen die Künstliche Intelligenz und Robotics betreffenden Entwicklungen.

8.1 Entwicklung bis zur Gegenwart

JAHR	ENTWICKLUNG
1854	George Boole entwickelt die Boolesche Algebra, mit der Logik in algebraischer Form dargestellt werden kann.
1921	Einführung des Wortes »Robot« durch den tschechischen Autor Karel Capek.
1942	Isaac Asimov veröffentlicht seine »Drei Gesetze für Roboter«.
1943	Colossus, der erste elektronische Rechner, wird in England fertiggestellt.
1948	Fertigstellung von Mark 1, dem ersten Computer, der in der Lage ist, ein Programm zu fahren.
1950	Der Mathematiker Alan Turing schlägt seinen später als »Turing Test« bekannten Test auf Künstliche Intelligenz vor.
1954	Erster großer Computer im Einsatz in der Industrie.
1956	John McCarthy prägt den Ausdruck der »Künstlichen Intelligenz« und organisiert den ersten KI-Kongreß.
1957	Der amerikanische Sprachwissenschaftler Noam Chomsky schafft mit seinem Werk »Syntactic Structures« die Basis für die heutige Spracherkennung.
1960	Entwicklung von LISP, der ersten KI-Sprache durch John McCarthy.
1960	GPS, der General Problem Solver, entwickelt von Allen Newell und Herbert Simon, ist in der Lage, Alltagsprobleme zu lösen.
1964	DENDRAL, ein chemisches Analyse-Expertensystem (siehe Teil 1, Expertensysteme), wird entwickelt.

1965 Noam Chomsky veröffentlicht sein Buch »Aspects of the theory of syntax«, das einen starken Einfluß auf spätere natürlichsprachige Systeme und maschinelle Sprachübersetzungs-Programme hat.

1966 Prof. Joseph Weizenbaum zeigt mit dem ELIZA-Programm, wie leicht ein Computer einen Psychiater imitieren kann.

1968 Der intelligente Computer HAL im Science-fiction-Film »2001« führt zu einer intensiven Diskussion über die Möglichkeiten Künstlicher Intelligenz in der breiten Bevölkerung.

1971 Mit dem Intel 4004 (4 bit) wird der erste Mikroprozessor geschaffen.

1972 Erste Prolog-Implementation durch Alain Colmerauer.

1972 Unimation, die erste Firma, die sich auf den Roboterbau spezialisiert, wird gegründet.

1972 SHRDLU, ein natürliche Sprache verarbeitendes System, wird von Terry Winograd am MIT entwickelt.

1973 Das erste kommerzielle Spracherkennungs-System kommt auf den Markt.

1980 Erster Micro-Mouse-Wettbewerb.

1981 Japan kündigt sein Projekt der fünften, intelligenten, Computergeneration an.

1981 Entdeckung von AIDS duch ein Experten-System.

1982 Die erste Phase des japanischen Großprojekts beginnt mit dem Ziel der Gewinnung neuer KI-Grundlagen.

1983 Das europäische ESPRIT-Programm, auf fünf Jahre mit einem Etat von 1,5 Milliarden Dollar angelegt, wird zur Förderung moderner Informationstechnologie ins Leben gerufen.

1983 Das HAM-RPM-System der Hamburger Universität ist fertiggestellt und genießt internationale Anerkennung. Es handelt sich um ein natürlichsprachiges System mit Vision-Verarbeitung.

1983 Erste preiswerte Vision-Systeme für Personalcomputer.

1983 Experten-Systeme auf Personalcomputern.

1984 Die erste Phase des japanischen Projekts wird erfolgreich abgeschlossen.

1985 Beginn der zweiten Phase des Fünfte-Generation-Programmes. Ziel ist die Verbesserung der Hardware-Struktur durch Entwicklung von Parallelprozessoren.

1985 Mit dem 1024-Kbyte-Atari 520 ST PLUS kommt eine neue Generation hochleistungsfähiger Personalcomputer auf den Markt.

8.2 Die chronologische Zukunftsentwicklung

Einige der voraussichtlichen Entwicklungen in der Zukunft sind im folgenden dargestellt:

JAHR	ENTWICKLUNG
1987	Erster »Robot Ping Pong Competition«.
1988	Intelligente Roboter der fünften Generation mit hoher Flexibilität ziehen verstärkt in Fabriken ein.

1988 Mega-Byte-Chip preiswert erhältlich.

1990 Der Kontrollraum der NASA in Houston, Texas, wird durch ein Experten-System ersetzt.

1990 Intelligente Lehrprogramme zu allen wichtigen Bereichen mit der Fähigkeit zum intelligenten Dialog und graphischer Unterstützung auf Homecomputern der neuesten Generation.

1992 Beendigung des Programms der fünften Computergeneration.

1995 Der Umsatz großer Computerfirmen wird fast ausschließlich auf Software basieren. IBM macht heute 20 Prozent seines Gewinnes mit Software, 1995 werden es voraussichtlich 80 Prozent sein.

Alle Angaben über die Zukunftsentwicklung beruhen auf realistischen Einschätzungen anerkannter Experten wie E. Feigenbaum, Christopher Evans, Pamela McCorduck, Peter Laurie sowie japanischer KI-Experten, die am Projekt der fünften Generation arbeiten.
Bedenken Sie auch, daß der Fortschritt in der »High Technology« allgemein exponentiell, nicht linear abläuft. Was uns genau in der Zukunft erwartet, läßt sich nur schwer abschätzen. Daher beschränkt sich die obige Liste lediglich auf als gesichert anzusehende Prognosen.

9 Das Forschungsprofil der Künstlichen Intelligenz und Robotics

Aus der anwendungsorientierten Sicht gliedert sich die KI in fünf große Teilbereiche:

1. Die Verarbeitung natürlicher Sprache
2. Experten-Systeme
3. Deduktions-Systeme
4. Robotertechnologie
5. Vision-Systeme

Auf diese Gebiete konzentrieren sich die heutige Forschung sowie die Lehre an Universitäten und Hochschulen. Ich möchte nun einen knappen, aber umfassenden Überblick über die einzelnen Teildisziplinen geben.

9.1 Verarbeitung natürlicher Sprache

Eine Definition des Gebietes lautet folgendermaßen:

»Die komplexen Informationsprozesse, die dem Verstehen, der Produktion und dem Erwerb natürlicher Sprache zugrunde liegen, sollen mit den Mitteln der Informatik exakt beschrieben und erklärt werden.«
(Prof. J. Siekmann)

Allgemein gesagt, soll die Kommunikation zwischen dem Menschen und der Maschine verbessert werden.

Das wichtige Teilgebiet der KI verfolgt im wesentlichen die folgenden drei Hauptaufgaben:

1. Verstehen von über die Tastatur eingegebenen Sätzen
2. Identifizierung echter natürlicher Sprache durch Schallmuster
3. Entwicklung neuartiger Dolmetscher-Programme

9.1.1 Verstehen geschriebener Sätze

Das typische Beispiel für ein System, das in der Lage ist, über die Tastatur eingegebene Sätze zu verstehen und auch richtig zu verarbeiten, ist das in diesem Buch vorgestellte Programm SHRDLU. Erinnern wir uns, das System leistet einiges, doch ist sein Sprachumfang nur auf die stark begrenzte Welt einer mit bunten Bauklötzchen besetzten Tischplatte eingeschränkt.
Heute versuchen KI-Forscher vor allem, Systeme zu entwickeln, die einen Diskurs über komplexere Gebiete zulassen. Der Forschungsschwerpunkt hat sich seit 1975 vom Problem des reinen Sprachverstehens auf die zusätzlichen Probleme des Dialogs verlagert.
Charakteristische Punkte sind die Rückführung des Dialogs auf spezielle Punkte, die unterschiedliche Motivation der Dialogpartner sowie die Fähigkeit, ein »Ausufern« des Gesprächs zu verhindern.
Die Problempunkte werden dadurch gelöst, daß das Computersystem während des Dialogs eine interne Repräsentation der ausgesprochenen Sachverhalte aufbaut. Dieser Vorgang ist das eigentliche Verstehen der gesprochenen Sprache. Mit Hilfe einer Wissensbasis (vgl. Kapitel über Experten-Systeme) werden dann sinnvolle Antworten über die angesprochenen Sachverhalte generiert.

9.1.2 Identifizierung gesprochener Sprache

Dieser KI-Bereich steckt noch in den Anfängen. KI-Forscher werden hier mit den unterschiedlichsten Problemen konfrontiert, die beim Erkennen von Schallmustern auftreten. Alle bisher entwickelten Systeme haben den Nachteil, daß der Anwender ungewohnt langsam sowie absolut klar und deutlich sprechen muß.
Das Hauptproblem liegt aber in Fremdsprechern. Man kann ohne weiteres ein leistungsfähiges System aufbauen, das einen einzigen Benutzer gut versteht. Doch bei anderen Sprechern mit einer anderen Betonung und Aussprache versagt ein derartiges System kläglich.
Der aktuelle Trend der KI-Forschung läuft momentan dahin, komplette Experten-Systeme zu erstellen, in deren Wissensbasis die unterschiedlichsten Sprachmuster verschiedener Anwender gespeichert sind, die das System selbständig ändern kann. Das ist jedoch mit einem sehr großen Aufwand verbunden.
Die Japaner entwickeln derzeit in der kürzlich begonnenen zweiten Phase ihres KI-Programms Prototypen von Systemen, die einen Wortschatz von 50000 Wörtern und ca. 100000 Ableitungsregeln mit einem prozentualen Genauigkeitswert von über 95 Prozent besitzen. Die 95 Prozent beziehen sich dabei auf eine Zahl von 100 Fremdsprechern.
Zusätzlich wird angestrebt, mit Hilfe synthetischer Sprache in Englisch und Japanisch einen intelligenten Dialog führen zu können.

9.1.3 Die Maschinenübersetzung

Bisher ergaben sich hier große Probleme – vor allem mit Doppeldeutigkeiten. DerWortschatz von automatischen Dolmetscher-Programmen ist stark begrenzt, und die Übersetzungsgenauigkeit beträgt momentan 75–80 Prozent. Die Japaner streben in diesem Bereich ein Maschinenübersetzungs-Programm an, das einen Wortschatz von 100000 Wörtern beherrscht und eine Übersetzungsgenauigkeit von über 90 Prozent besitzt.

9.1.4 Die Anwendungen der Übersetzer

Die unmittelbaren Anwendungen, die heute bereits angestrebt werden, liegen in der Kopplung einer natürlichsprachigen Schnittstelle mit einer Datenbank, einem Experten-System, einem Informations-System oder einer Roboterkontrolle.
Die Forschung war bisher sehr erfolgreich. Das führte zu erheblichen Investitionen amerikanischer und japanischer Firmen und Forschungszentren.
Man darf auf die Ergebnisse der sehr ehrgeizigen Ansätze gespannt sein. Sie werden noch in diesem Jahrzehnt erwartet.

9.2 Experten-Systeme

Knapp zusammengefaßt, sollen in diesem KI-Gebiet Programme entwickelt werden, die Aufgaben erfüllen, die bisher menschlichen Spezialisten vorbehalten waren. Das Anwendungsspektrum ist schon heute vielfältig. Die Schwerpunkte liegen auf den folgenden Gebieten:

- Allzweck-Tools:
  Systeme zur Entwicklung von Experten-Systemen, Wissenserwerb-Systemen, Ableitungs-Systemen sowie Diagnose-Systemen
- Betriebswissenschaft:
  Risiko- und Kostenbewertung beispielsweise von Bauprojekten, Datenverwaltungs-Systemen
- Biotechnik:
  Strukturanalyse und Synthese von DNS
- Chemische Industrie:
  Organische Synthese, Strukturanalyse
- Computer-Systeme:
  Diagnose von Computer-Defekten, Verkaufsberater von Computer-Systemen
- Computing:
  Experten-Systeme zur Unterstützung von Software-Entwicklung, automatische Programmierung

- Erziehung:
  Computergestützte Unterrichts-Systeme (CAI), Lehr-Systeme für Computersprachen
- Herstellung:
  Überwachung von Arbeitsabläufen, Verwaltung von Großprojekten, Fabrik-Arbeitsabläufe und Robotics
- Medizin:
  Infektionen, Diagnose-Systeme, Therapiediagnose und Hypothesen, Krebsbehandlung, Intensivpflege-Systeme
- Militär:
  Erkennung und Verfolgung von Schiffen und U-Booten, taktische Zielführungen in der Luft, Analyse strategischer Indikatoren und Warnungen, taktische Kampffeld-Kommunikation
- Recht:
  Assistent für Rechtsanwälte, Steuerrechtsgesetze
- Rohstoffsuche:
  Ölsuche, Wasserressourcen-Probleme, Minerallager
- Technik:
  VLSI-Entwicklung, Entwicklung neuartiger dreidimensionaler Mikrochips, Steuerung von Atomreaktoren
- Wissenschaft:
  Planung von Genspaltungsversuchen

Anmerkung:
Unter jedem Gebiet wurden einige Anwendungsbereiche aufgeführt, in denen schon heute Experten-Systeme erfolgreich operieren. Experten-Systeme demonstrieren, wie weit es bereits gelungen ist, Fähigkeiten auf dem Computer zu realisieren, die bisher nur der menschlichen Intelligenz vorbehalten waren.

9.3 Deduktions-Systeme

Ursprünglich wurde auf diesem Gebiet hauptsächlich das Beweisen mathematischer Sätze und Theoreme behandelt. Doch aus dem rein mathematischen Bereich wurden Anwendungen in der Informatik gefunden, wie:

- Logik als Programmiersprache
- Programmsynthese
- Programmverifikation
- Beweis der Fehlerfreiheit integrierter Schaltkreise
- Steuerung von Atomreaktoren
- Verwaltung allgemeiner Organisations-Strukturen

Die höchste Priorität liegt derzeit bei der Problematik der Programmverifikation, der Überprüfung eines Algorithmus auf seine Fehlerfreiheit.

9.4 Der Bereich der Robotics

Das Ziel des immer mehr an Bedeutung gewinnenden KI-Gebietes ist die Entwicklung von Robotern mit Eigenintelligenz.
Die Forschung im Bereich der Robotics ist stark anwendungsorientiert, Erkenntnisse werden meistens sofort kommerziell angewendet. Schwerpunkt ist derzeit die Entwicklung flexibler, mit umfassenden Sensoren ausgestatteter Roboter.
Die Flexibilisierung von Robotern läuft darauf hinaus, daß ein robotergesteuerter Produktionsablauf bei Bedarf nach Belieben geändert werden kann.
Roboter sind heute bereits in der Lage, sich zum einen an die Umwelt anzupassen, zum anderen eine gesamte Produktionspalette in kurzer Zeit zu ändern. Jedes Werkstück kann unterschiedlich behandelt werden. General Motors hat Roboter entwickelt, die z.B. je nach Belieben einzelne Autos in den unterschiedlichsten Farben und Ausstattungen herstellen können. Das kann so aussehen, daß zuerst ein rotes Auto mit zwei Türen aus der Produktion läuft, dann ein grünes viertüriges sowie ein weiteres mit einem Schiebedach usw.
Entscheidend ist auch die Mechanik, vor allem die maximale Arbeitsgeschwindigkeit sowie die Präzision des Roboters, verbunden mit der Fähigkeit zu »Real Time«-Aktionen.
Im Trend liegen außerdem

- robotergerechte Konstruktion (CAD/CAM)
- graphische Simulation
- Mensch-Maschine-Schnittstelle (z.B. in natürlicher Sprache)
- Experten-Systeme und Robotics
- Neue Anwendungsbereiche wie Raumfahrt und Home-Roboter in Dienstleistungsbereichen
- Autonome Roboter

Weitere Entwicklungstendenzen, wie der Einsatz moderner Lasertechnologien, und neuartige Sensortechnik wurden bereits im Roboterkapitel erörtert. Hand in Hand mit Robotics läuft die Entwicklung von Vision-Systemen.

9.5 Das Bild-Verstehen

Neben der Verarbeitung natürlicher Sprache ist dieses Gebiet sicher eines der größten und wichtigsten Unternehmen der KI-Forschung. Die Bildverarbeitung nimmt einen derart großen Bereich ein, daß selbst Spezialisten das Gebiet nicht mehr im einzelnen überblicken können.

Schwerpunkt ist die Gestaltwahrnehmung. Sie ist bereits im zweidimensionalen Bereich problematisch genug, doch ein Teilbereich beschäftigt sich auch mit der dreidimensionalen Auswertung von Bildern. Hauptinteressent ist die U.S. Air Force, die auf diese Weise Luftbildaufnahmen von Flugzeugen und »Spionagesatelliten« noch effektiver auswerten will.
Auch das sehr ehrgeizige Bilderkennungs-System der Japaner mit einer Kapazität von 100000 Bildern wurde bereits erwähnt.

Vision-Anwendungsbereiche

Die Anwendungsbereiche sind vielfältig. Die wichtigsten Gebiete sind:

- Robotics, sehende intelligente Roboter
- medizinische Anwendungen, wie Reihenuntersuchungen von Röntgenaufnahmen
- Auswertung von Luftbildaufnahmen
- Vergleich von Photos
- Klarschriftleser

Das Gebiet ist sicher sehr interessant, einige der vielen möglichen Anwendungen beinhalten aber auch Risiken.
Ein Beispiel für mögliche risikoreiche Faktoren der Vision-Systeme ist das HAM-RPM-System. Das an der Hamburger Universität entwickelte System ist in der Lage, beispielsweise einen Straßenzug längere Zeit zu beobachten. Dabei merkt es sich alle Einzelheiten, auf die es programmiert wurde.
Anschließend kann der Anwender in einem natürlichsprachigen Dialog alle Einzelheiten abrufen, indem er Fragen stellt, wie

»Ist in dem Zeitraum von x bis y Uhr ein Mann mit einem Regenmantel auf dem rechten Bürgersteig vorbeigelaufen?«

Hier deutet sich die Möglichkeit einer Überwachung an. Derartige Risiken der Bildverarbeitung sind vergleichbar mit denen der Genforschung. Um bedenklichen Entwicklungen vorzubeugen, sollte die Anwendung der Forschung kontrolliert werden. Nicht alles mittels der KI-Techniken Machbare sollte auch in der Praxis realisiert werden.

9.6 Ein Fazit

Zusammenfassend kann man sagen, daß unter allen KI-Fachleuten Einigkeit darüber herrscht, daß der Künstlichen Intelligenz eine Schlüsselfunktion beim Einsatz von Computern im kommenden Jahrzehnt zufällt. Dies liegt vor allem an der, in diesem Kapitel teilweise aufgezeigten, großen Anwendungsvielfalt. Zudem lassen sich wissenschaftliche Ergebnisse sofort praktisch anwenden. Viele der von der KI erzielten wissenschaftlichen Resultate werden in den neunziger Jahren eine entscheidende Rolle in der Computer-Technologie einnehmen.

»KI wird zu einem wichtigen Bestandteil technologischer Wettbewerbsfähigkeit.« (Prof. E. Feigenbaum).

Bei dem Stellenwert, der der KI zugemessen wird, ist es auch für Sie sicher interessant, sich weiter über dieses Gebiet zu informieren. Daher erhalten Sie in Kapitel 12 eine Zusammenstellung der wichtigsten Basisliteratur zur KI.

10 Die Perspektiven der Künstlichen Intelligenz

Alle Voraussagen sind schwierig, vor allem solche über die Zukunft. Dieser Satz wird dem Physiker Nils Bohr zugeschrieben. Er trifft im besonderen auf das Forschungsgebiet der Künstlichen Intelligenz zu.
Werfen wir an dieser Stelle einen Blick auf die Gegenwart der KI. Derzeit werden immer mehr Ergebnisse der KI-Forschung auf andere Gebiete des Computerbereichs übertragen. Die Erkenntnis, daß sich KI-Techniken gewinnbringend und effektiv auf ein breites Spektrum von Einsatzbereichen anwenden lassen, hat sich fast überall durchgesetzt.
Die ersten Ergebnisse dieser Anschauung lassen sich bereits im Home- und PC-Computer-Bereich aufzeigen. So entstammt beispielsweise die Fenstertechnik wie GEM und die MAUS-Steuerung der KI-Forschung. Ziel war dabei die Entwicklung einer möglichst effektiven Benutzeroberfläche.
Heute gehören WINDOWS und die MAUS schon fast zum Standard von Personalcomputern. Die beiden Errungenschaften sind nur ein Nebeneffekt der KI-Forschung. Aber sie machen deutlich, welche entscheidenden Anstöße und vorteilhaften Entwicklungen von der KI-Forschung ausgehen.
Richtig interessant wird es aber bei der Integration »Künstlicher Intelligenz« in PC-Software. So entstammen auch Programme wie »Ideenprozessoren« der KI-Entwicklung. Es gibt auch bereits Programme zur Sprachübersetzung vom Deutschen ins Englische und Französische für PCs.
Fortgeschrittene Compiler wie der PQCC, der »Production Quality Compiler Compiler«, die einen optimierten Code generieren, basieren auf einer Ansammlung von individuellen Experten-Modulen. Diese beinhalten das Wissen zur Algorithmus- und Code-Optimierung.
Für Homecomputer wurden Ende 1985 in den USA neuartige intelligente Adventures vorgestellt. Sie sind fähig, normale englischsprachige Eingaben zu verstehen. Bisher war das Vokabular von Adventures auf einfache Anweisungen wie

GO NORTH

oder

EXAMINE ROOM

beschränkt. Die neue Adventure-Generation ist nun in der Lage, auch komplexe Eingaben zu verstehen und alle gegebenen Anweisungen auszuführen.
Ein eingebener Beispielsatz lautet folgendermaßen:

> Go north and follow the path up to the top of the hill. If you see something unusual then stop.

Auch noch kompliziertere Eingaben als dieses Beispiel werden durch ein intelligentes Parsing-Modul erfaßt.
Es gibt auch Experten-Systeme für professionelle Anwendungen auf Personalcomputern, die als intelligente Assistenten eingesetzt werden können. Die universellen Möglichkeiten von Experten-Systemen wurden ausführlich in diesem Buch vorgestellt.
Großen Einsatz findet die KI derzeit im Database-Management. Intelligente Datenbanken suchen hier anhand unterschiedlichster Informationen nach spezifischen Daten. Dabei wird vor allem an einer Benutzerschnittstelle in natürlicher Sprache gearbeitet.
Ein Beispiel dafür, was auf dem Gebiet der Verarbeitung von natürlicher Sprache möglich ist, stellt das HAM-RPM-Programm dar. Es simuliert das sprachliche Verhalten eines Hotelmanagers. Dabei versucht es, eine möglichst positive Darstellung des Hotels zu erzeugen und ein Zimmer zu vermieten.
Ein wichtiges Gebiet, auf dem KI-Techniken angewendet werden, stellt die Automatische Programmierung dar. Derartige Programme sind auch bereits für PCs erhältlich. Nach Eingabe einer Problembeschreibung wird das Problem automatisch in einen Algorithmus umgesetzt und in einer normalen Programmiersprache formuliert.
Einen weiteren vor allem in der Zukunft sicher immer wichtiger werdenden Bereich der Anwendung von KI-Techniken stellt die CAI dar, die »Computer Aided Instruction«. Intelligente Lehr- und Lernprogramme, die mit dem Schüler in natürlicher Sprache kommunizieren und Wissen anschaulich graphisch darstellen, existieren bereits heute auf teuren Computer-Systemen. Doch mit der neuen Computer-Generation wie dem ATARI ST mit 1 Mbyte RAM können derartige Systeme auch für einen breiten Interessentenkreis preiswert angeboten werden.
Die Systeme können sich dem Leistungsstand des Schülers genau anpassen und ihn individuell betreuen. Im interaktiven Dialog werden die Schwächen und Stärken des Schülers ermittelt und genau berücksichtigt. Wissen wird so sehr viel schneller und effektiver als mit den herkömmlichen Lehrmethoden vermittelt.
Damit befinden wir uns in unseren Betrachtungen zur CAI bereits in der nahen Zukunft. Wie wird sich die KI weiterentwickeln? Die bereits heute existierenden intelligenten Datenbanken und die CAI werden mit den nun vorhandenen Speichergrößen preiswerter Computer im Mbyte-RAM-Bereich sicher eine immer stärkere Verbreitung finden.
Die Systeme, die größtenteils bereits heute einsatzfähig in den KI-Labors bereitstehen, werden dazu führen, daß jedem Anwender ein Zugang zu sinnvoll ausgewähltem und klug organisiertem Wissen möglich ist.
Das klingt recht harmlos, es handelt sich dabei allerdings um ein Ereignis, dessen Tragweite sich heute noch gar nicht absehen läßt. Die tiefgreifendste Veränderung wird wahrscheinlich in der in naher Zukunft liegenden automatischen Erzeugung von Wissen eintreten:
Die automatische Wissensgenerierung kann unberechenbare Auswirkungen haben. Wir haben

heute keinerlei Vorstellungen davon, was passieren kann, wenn ein Computer alles ihm zur Verfügung stehende Wissen verwenden kann. Was geschieht, wenn das System die Wissensverarbeitung auf eine systematische Weise vornimmt, die dem Menschen nicht möglich ist?

Der Mensch ist durch seinen Evolutionserbteil auf etwa vier Themen beschränkt, die maximal gleichzeitig verarbeitet werden können. Ein Computer besitzt diese Einschränkung nicht. Besonders durch Parallelrechner werden atemberaubende Rechengeschwindigkeiten erzielt. Was passiert, wenn derartige Rechner bei der Wissensverarbeitung weiterreichende Folgerungen als die Menschen ziehen können?

In der Vergangenheit hat sich gezeigt, daß wir mit der Einführung einer neuen Technik verlernen, bestimmte Dinge zu tun. In der Schule haben wir gelernt, wie man schriftlich dividiert, multipliziert, Wurzeln zieht oder Logarithmen bildet. Heute können das nur noch die wenigsten Menschen. Die meisten überlassen diese Arbeit Taschenrechnern. Es ist gut möglich, daß Computer uns in Zukunft weiterreichende Arbeiten abnehmen und uns mehr Zeit für komplexere Problemlösungen lassen.

Es ist allerdings ungewiß, ob ein Computer, der schneller und weitreichender als Menschen denken kann, nicht auf ganz anderen Bahnen denkt und zu neuen Schlußfolgerungen kommt.

Doch kehren wir nun in die Gegenwart zurück.

In diesem Buch wurde Ihnen eine umfassende, praxisbezogene Einführung in den faszinierenden Bereich der Künstlichen Intelligenz und Robotics vermittelt. Ich möchte Sie an dieser Stelle dazu anregen, mit den vorgestellten Techniken und Programmen zu experimentieren und Künstliche Intelligenz in Ihren eigenen Programmen zu verwenden.

11 Glossar der Künstlichen Intelligenz

Im nachfolgenden Wörterbuch werden die wichtigsten Begriffe aus dem Bereich der Künstlichen Intelligenz herausgegriffen und näher erläutert.
Den größten Stellenwert nehmen dabei englische Fachausdrücke ein. Die jeweiligen Erklärungen sollen Ihnen helfen, mit weiterführender KI-Literatur, die größtenteils englischsprachig ist, besser zurechtzukommen. Die Zusammenstellung enthält auch Begriffe, die im Rahmen dieses Buches des besseren Verständnisses wegen nicht erwähnt wurden.

AI

Artificial Intelligence = Künstliche Intelligenz.

ALPHA-BETA PRUNING

Suchmethode in einem Problembaum, bei der jeweils einzelne Baumknoten (Zustände eines Problems) eliminiert und vom Suchprozeß ausgeklammert werden. Die Methode stellt eine Variation des → Minimax-Search-Verfahrens dar.

ATN

Argumented Transition Network, ein erweitertes Übergangs-Netzwerk. Es wird in der Verarbeitung natürlicher Sprachen als Repräsentationsmethode für Grammatiken eingesetzt.

A* ALGORITHMUS

Er stellt einen Suchalgorithmus zur Implementierung heuristischer Bewertungsfunktionen dar.

B* ALGORITHMUS

Algorithmus, der mit normalen → problem solving trees und → game trees Spiel-Probleme löst.

BACKTRACKING

Vertikales Durchsuchen eines Problembaumes (→ Depth-first-Verfahren).

BACKWARD CHAINING

Ein denkendes System, das von einigen selbst vorgenommenen Annahmen ausgeht und dann ermittelt, ob sie wahr sind oder nicht.

BACKWARD REASONING

Hier wird von der Lösung eines Problems ausgegangen und versucht, den Ausgangszustand zu erreichen. Ein Problem wird dabei von »hinten nach vorne« analysiert.

BELIEF SYSTEMS

Belief Systems gehen von einem Satz an Annahmen aus, die durch Beweise zu einer Lösung führen.

BEST FIRST SEARCH

Ein Ordnungsprinzip sowohl bei der Abarbeitung als auch bei der Erweiterung eines Suchbaumes. Dabei werden die Faktoren der höchsten Priorität zuerst berücksichtigt.

BIDIRECTIONAL SEARCH

Abarbeiten eines Baumes gleichzeitig von der Wurzel und der Spitze. Anders ausgedrückt, ein Problem wird gleichzeitig vom Ausgangszustand und vom Zielzustand gelöst.

BLIND SEARCH

Blinde Suche nach Zusammenhängen. Es werden keine Suchkriterien (heuristische Methoden) angewendet.

BRANCH AND BOUND

Bei dieser Baum-Suchstrategie werden die jeweils kürzesten Lösungswege für die Teilprobleme ermittelt und weiterverfolgt.

BRANCHING FACTOR

Er gibt für die momentan erreichte Ebene eines Suchbaumes die durchschnittliche Zahl der folgenden Problemebenen (Knoten) an. Er ist damit ein Maß für die benötigte Suchzeit.

BREADTH-FIRST SEARCH

Vgl. Depth-First Search. Die Durchsuchung eines Problembaumes erfolgt hier aber horizontal anstatt vertikal.

COMBINATORIAL EXPLOSION

Mit jeder weiteren Ebene eines Baumes wächst er in seinen Faktoren exponentiell an, bedingt durch kombinatorische Regeln der Wahrscheinlichkeitsrechnung.

CONTROL STRATEGY

Die Kontrollstrategie ist ein Steuerungsverfahren zur Lösung eines Problems. Sie legt fest, welche einzelnen Operatoren wann auf einzelne Daten angewendet werden.

Man unterscheidet drei Kontrollstrategien:

data driven – abhängig vom Dateninhalt;

irrevocable – unwiderruflich, der gewählte Suchweg muß auf jeden Fall verfolgt werden;

tentative – versuchsweise, dabei können einmal gewählte Wege rückgängig gemacht werden.

DATABASE
Datenbank, in ihr sind alle Ausgangsdaten, die zur Lösung eines Problems benötigt werden, enthalten.

DATA DRIVEN STRATEGY
Siehe Forward Chaining.

DATA STRUCTURE
Die Form, in welcher die → Knowledge Base strukturiert ist → Knowledge Representation.

DEDUCTION
Logische Herleitung eines Problems.

DEDUCTION SYSTEMS
Deduktions-Systeme werden vor allem angewendet, um mathematische Sätze zu beweisen und die Verifikation von Programmen vorzunehmen. Diese KI-Disziplin nimmt innerhalb der heutigen Forschung einen immer höheren Stellenwert ein.

DEGREE OF A TREE
Der Grad stellt die Ordnung eines Suchbaumes, also die Anzahl seiner Stufen/Ebenen, dar.

DEPTH BOUND
Die maximale Suchtiefe innerhalb eines Suchbaumes, wichtig als Abbruchbedingung eines Suchvorgangs.

DEPTH-FIRST SEARCH
Vertikales Durchsuchen eines Problembaumes.

DERIVATION TREE
In diesem Baum werden Terme dargestellt, die von einem Ausgangsterm abgeleitet werden.

DOMAIN OF ENQUIRY
Das Gebiet, in dem ein Experten-System Spezialist ist, z.B. Ölsuche anhand geologischer Faktoren.

ES
Experten-System.

EVALUATION FUNCTION
Die Funktion weist einem Knoten, einem Operator eines Baumes eine Zahl zu, deren Wert ein Kriterium für die Ordnung, also die Qualität des Operators, ist.

EXPANSION OF A NODE
Stellt die Erweiterung eines Baumes um einen Operator dar. Vgl. EXPERTENSYSTEM-Baumerweiterung im Lern-Modus.

EXPERT SYSTEM

Ein intelligenter Assistent für ein begrenztes Spezialgebiet, auf dem er gestellte Probleme anhand einer Datenbank lösen kann.

FORMAL LANGUAGE

Eine formale Sprache ist eine Sprache, die mit wenigen, aber prägnanten syntaktischen Regeln auskommt. BASIC, Pascal und C sind formale Sprachen.

FORWARD CHAINING

Ein intelligentes System, das am Anfang eines Problems beginnt und dann versucht, es so gut es kann zu lösen. Es ist wesentlich einfacher zu programmieren als ein Backward reasoning/chaining-System.

FRAME

Frames werden verwendet, um eine Anzahl Attribute zu einem bestimmten Objekt, wie einem Stuhl, zu beschreiben. Vgl. → Script.

FRAMEWORK

Grundgerüst; die grundlegende Struktur eines Programms. → Shells bilden ein Framework für ein Experten-System.

FULL-WIDTH SEARCH

Hier wird die Suche nach Kriterien im gesamten Problembaum vorgenommen. Die Suchmethode erfordert sehr viel Zeit. Daher wird meistens versucht, das Problem auf wenige Zweige eines Baumes einzugrenzen.

GAME TREE

Der Suchbaum für ein Spiel, wird z.B. bei Schach eingesetzt.

GENERATE AND TEST

Ein System, das eine Lösung eines Problems generiert und dann überprüft, ob sie korrekt ist. Dies ist nützlich, wenn die gesamte Anzahl aller Lösungen nicht in den Speicher eines Computers paßt. Statt alle möglichen Lösungen ständig zugriffsbereit gespeichert zu haben, werden sie ständig neu generiert.

GOAL STATE

Zielzustand, d.h. Lösung eines Problems.

GPS

General Problem Solver, ein KI-Programm, das alltägliche Probleme löst.

HEURISTIC SEARCH

Bei einer heuristischen Suche wird ein Problem gelöst, indem alle vorhandenen Informationen zur Ermittlung der Lösung berücksichtigt werden.

HEURISTIK

Eine Regel oder Prozedur, die ein Problem durch ein intuitives Verfahren löst.

HILL CLIMBING SEARCH

In einem Suchbaum wird von dem Knotenpunkt, an dem sich das Programm momentan befindet, der höchstwertige Knoten in der Umgebung ausgewählt. Die Suche wird dabei in einer Richtung durchgeführt (→ Best Search, → Ordered Search).

HYBRID CONTROL STRATEGY

Im Gegensatz zur normalen → Control Strategy wird hier mit verschiedenen Kontrollverfahren gearbeitet.

INFERENCE ENGINE

Bezeichnung für ein Programm, das die → Knowledge Base steuert. Sie ist in der Lage, ähnlich aufgebaute Knowledge Bases mit unterschiedlichem Inhalt zu verwalten.

INITIAL STATE

Anfangszustand eines Problems.

INTELLIGENT ASSISTENT

Experten-System.

INTELLIGENT KNOWLEDGE BASED SYSTEM

Ein anderer Ausdruck für Experten-Systeme.

INTERLISP

Lisp-Version, die die Vorzüge von → Lisp mit denen von → Prolog vereinigt. Der Dialekt wird z.B. von Siemens, Nixdorf und Xerox eingesetzt.

KB

Siehe Knowledge Base.

KIPS

Wissens- und Informationsverarbeitungssysteme, die neue, fünfte Generation der Computer-Systeme, wie sie die Japaner momentan entwickeln. Sie besitzen hohe Verarbeitungsgeschwindigkeiten, hervorragende Schnittstellen zum Menschen und sind mit sehr großen Wissensbanken verbunden.

KNOWLEDGE ACQUISITION

Sie stellt den Vorgang des Auffüllens der → Knowledge Base dar. Das kann entweder manuell oder durch selbstlernende Systeme geschehen.

KNOWLEDGE BASE

Sie enthält alle Daten, die das Experten-System verwendet. Die Daten sollten transparent konzipiert sein, d.h., der User soll die Möglichkeit haben, das Wissen zu kontrollieren und festzustellen, ob es sinnvoll ist. Dies ist wichtig zur Pflege und Erweiterung des Wissens.

KNOWLEDGE ENGINEER

Der Mensch, der meist mit Hilfe eines Spezialisten eines Fachgebietes das Wissen des Experten in computergerechte Regeln umwandelt und diese zu einem Experten-System zusammensetzt.

KNOWLEDGE REPRESENTATION

Darstellung des Wissens innerhalb des Computers → Data Structure.

LISP

Abkürzung für LIS-t P-rocessing Language. Mit → Prolog die beliebteste KI-Programmiersprache. Sie ist speziell zur Verarbeitung von Datenlisten konzipiert.

MEAN ENDS ANALYSIS

Ein Problem wird hier gelöst, indem der aktuelle Zustand eines Problems jeweils mit dem Zielzustand verglichen wird. Dabei versucht das KI-Programm, die Problemstellung durch schrittweise Reduzierung der Differenz der beiden Zustände zu lösen.

MINIMAX SEARCH

Spezielles Suchverfahren für Spiel-Bäume. Dabei wird, um abwechselnd Züge des Computers und des Spielers zu simulieren, jeweils ein hochwertiger und ein niedrigwertiger Knoten ausgewählt.

NATURAL LANGUAGE PROCESSING

Verarbeitung natürlicher Sprache, vgl. auch die ersten Kapitel dieses Buches.

NETWORK

Ein Netz, also Computer und Kommunikationsverbindungen, ermöglicht Rechnern, miteinander Verbindung zu halten und Daten, Wissensbänke oder Programme auszutauschen. Man unterscheidet zwischen lokalen, landesweiten und globalen Netzen.

NODE

Knoten eines Baumes, d.h. ein bestimmter Zustand eines Problems.

NONMOTONIC REASONING

Traditionelle Systeme sind nonmotonic, d.h. die Anzahl der als wahr bekannten Regeln nimmt stetig zu. Dadurch werden alte Regeln aber nicht falsch.

ORDERED SEARCH

Bei der geordneten Suche wird die Abfolge der zu wählenden Knoten anhand einer oder mehrerer spezieller Bewertungsfunktionen bestimmt.

PARSING

Analyse eines natürlichsprachigen Satzes durch Erkennung und Zuweisung seiner syntaktischen und semantischen Struktur.

PATTERN MATCHING

Musteranpassung ist die Methode, um Übereinstimmungen zwischen Problemen und gespeicherten Daten zu ermitteln.

PATTERN RECOGNITION

Die Mustererkennung wird vor allem bei Bild- und Sprachanalysen, also zur Erkennung regelmäßiger Strukturen, angewendet.

PERCEPTION

KI-Bereich der Wahrnehmung:

Vision – optische Wahrnehmung
Speech – Spracherkennung

PLANNING

Die Planung wird zur Lösung komplexer Probleme, besonders in Verbindung mit Robotics, verwendet.

PROBLEM REDUCTION

Einer der wesentlichen Bereiche der KI-Forschung. Durch Zerlegung eines Problems in Teil- und Unterprobleme bzw. einfach lösbare Aufgaben können auch die komplexesten Aufgaben sinnvoll gelöst werden.

PROBLEM REDUCTION REPRESENTATION

Ähnlich wie bei der → Knowledge Representation wird hier die interne Problemdarstellung behandelt.

PROBLEM REPRESENTATION

Interne Problemdarstellung im Computer.

PROBLEM SOLVING

Problemlösung.

PRODUCTION RULES

Die → Knowledge Base eines → ES wird durch Production Rules dargestellt. Die Regeln haben oft die »Wenn-dann«-Form.

PROGRAM GENERATORS

KI-Programme, die selbständig anhand einer kurzen Problembeschreibung ein komplettes Programm in einer → formalen Programmiersprache erstellen.

PROLOG

Die KI-Programmiersprache findet ihre Anwendung vor allem in der Entwicklung von Experten-Systemen. Bekannt und populär ist Prolog hauptsächlich durch die Entscheidung der Japaner geworden, die Sprache als Basis ihres Fünfte-Generation-Projekts zu wählen.

REASONING

Aufgrund eines Urteils wird über eine Beweisführung ein logischer Schluß gezogen.

ROBOTICS

Spezialgebiet der KI. In Zusammenarbeit mit der KI werden autonome Roboter entwikkelt, die sehen, fühlen und mittels Software intelligente Entscheidungen treffen können.

RULE BASE

Die Regelbasis beinhaltet die Menge aller Operatoren, die auf die Daten und die Entscheidungsknoten eines Baumes wirken.

RULE-BASED SYSTEM

Ein Experten-System, das seine Entscheidungen stark auf Production Rules stützt.

RULE VALUES

Jede Regel erzeugt einen spezifischen Wert für ein bestimmtes Datenmuster. Die Regel mit dem höchsten Wert wird als nächste behandelt.

RULE VALUE SYSTEMS

Diese Systeme operieren, indem sie versuchen, das Maß an Ungewißheit, unter dem ein Experten-System leidet, zu reduzieren.

SCRIPT

Ein Script beschreibt eine Sequenz von Vorgängen.

SEARCH ALGORITHM

Suchalgorithmus.

SEARCH GRAPH

Suchgraph, Teil des Problembaumes.

SEARCH TREE

Suchbaum.

SEARCH SPACE

Suchraum eines Suchalgorithmus.

SHELL

Eine Experten-System-Shell ist ein Rahmen, in dem ein Experten-System schnell erstellt werden kann, indem die einzelnen Regeln und damit das spezifische Wissen nachträglich implementiert werden.

Es besteht meistens aus einer Inference Engine und einem Programm-Modul, das die Knowledge Base festlegt. Die Shell ähnelt Programm-Generatoren; anstelle von Programmen werden jedoch Experten-Systeme erstellt.

SHRDLU

KI-Programm, das mit dem Anwender einen intelligenten Dialog über eine Klötzchenwelt auf einer Tischplatte führt und Tätigkeiten ausübt.

SOLUTION TREE

Der Lösungsbaum ist in einem Teil des Suchbaums enthalten und dient zur Festlegung des Lösungsweges.

STATE

Zustand des Problems, gibt die Problemebene wieder.

STATE SPACE

Der Zustandsraum beinhaltet die Menge der möglichen Zustände, die ein Problem annehmen kann.

SUPERVISOR

Programmkopf/Leiter, der Steuerungsteil eines Programms.

THEOREM PROVING

Der Theorembeweis stellt einen wichtigen Teil bei → Deduction Systems dar.

TREE

Baum.

TURING TEST

Dieser Test ist der von allen Wissenschaftlern anerkannte Test auf wirkliche Intelligenz durch natürlichsprachige Kommunikation.

VERIFICATION

Überprüfung von Programmen im Bereich der → Deduction Systems.

VLSI

Hohe Integration von Transistoren auf Chips. Die modernsten Chips enthalten maximal eine halbe Million Transistoren. Der sich in der Entwicklungsphase befindende Mega-Chip soll zehn Millionen Transistoren enthalten.

12 Basisliteratur zur Künstlichen Intelligenz

Wenn Sie weiterführende Literatur studieren wollen und damit weiter in das interessante, aber auch komplexe Gebiet der Künstlichen Intelligenz einsteigen wollen, werden Sie schnell feststellen, daß es eine fast unüberschaubare Fülle an Literatur gibt.
Um Ihnen eine Hilfe bei der Auswahl der Literatur zu den speziellen Fachgebieten zu geben, wird in diesem Kapitel ausgewählte KI-Literatur vorgestellt. Fast alle Fachbücher sind in Englisch geschrieben. Deutsche KI-Fachbücher sind rar.

12.1 Einführende Lehrbücher

A.Barr, E.Feigenbaum »The Handbook of Artificial Intelligence«, vol. I, vol. II, vol. III, William Kaufman Inc. 1981

Feigenbaum and Feldman »Computers and Thought«, McGraw Hill Inc.

N.Nilson »Problem Solving Methods in Artificial Intelligence«, McGraw Hill, New York, 1971

N.Nilson »Principles of Artificial Intelligence«, Springer, 1982

B.Raphel »The Thinking Computer: Mind inside Matter«, San Francisco, W.H. Freeman, 1976

Elaine Rich »Artificial Intelligence«, McGraw Hill Book Inc., 1984

P.H. Winston »Artificial Intelligence«, Addison Wesley Publ. Company, 1977

12.2 Verarbeitung natürlicher Sprache

E.Charniak, Y.A. Wilks »Computational Semantics«, North Holland, 1976

H.Tennant »Natural Language Processing«, vol. I: Syntax, Addison Wesley Publ. Comp., 1983

D.Walker »Understanding Spoken Language«, Elsevier North Holland, 1978

T.Winograd »Language as a cognitive Process«, vol. I: Syntax, Addison Wesley Publ. Comp., 1983

12.3 Automatische Beweise

W.Bibel »Automated Theorem Proving«, Vieweg, 1982

C.Chang, R.Lee »Symbolic Logic and Mecanical Theorem Proving«, Academic Press, 1973

12.4 Computer-Vision

Faugeras »Fundamentals in Computer Vision«, Cambridge University Press, 1983

E.L. Hall »Computer Image Processing and Recognition«, Academic Press, 1979

A.R. Hanson, E.M. Riseman »Computer Vision Systems«, Academic Press, 1978

P.H.Winston »The Psychology of Computer Vision« McGraw Hill, 1975

12.5 Experten-Systeme

D.B.Lenat »Knowledge-based Systems in Artificial Intelligence«, McGraw Hill, New York, 1982

D.Michie »Introductory Readings in Expert Systems«, Gordon and Breach, New York, 1982

D.Michie »Expert Systems in the Microelectronic Age«, Edinburgh University Press, 1979

12.6 KI-Sprachen

J.Allen »Anatomy of LISP«, McGraw Hill, 1978

W.F.Clocksin, C.S. Mellish »Programming in Prolog«, SpringerVerlag, 1984

Ennals »Beginning in micro-Prolog« Chichester: Ellis Horwood, 1982

H.Stoyan, G.Görz »LISP - Eine Einführung in die Programmierung«, SpringerVerlag, 1984

Anhang: Bibliographie zur Künstlichen Intelligenz

Das Literaturangebot für den Themenbereich der Künstlichen Intelligenz und der Robotics hat einen großen Umfang angenommen. Leider nehmen deutschsprachige Werke nur einen kleinen Teil der Veröffentlichungen ein.
Das folgende Verzeichnis bietet Ihnen die Möglichkeit, sich spezielle Informationen zu einem Themenbereich der KI zu verschaffen.

A. Englischsprachige Fachliteratur

Albus, »Brains, Behavior and Robotics«, McGraw Hill 1981, New York
Barr, Feigenbaum, »The Handbook of Artificial Intelligence«, Los Altmos, Kaufmann, 1981
Boic, »Natural Language Communication with Computers«, Springer, 1978
Carness, »Human Chess Skill«, Springer, 1977
Dreyfus, »What Computers Can't Do«, Harper and Row, New York, 1979
Elcock, Minchie, »Machine Intelligence 2«, Chichester, Ellis Horwood, 1977
Feigenbaum, »Computer and Thought«, McGraw Hill, New York, 1963
Frey, »Chess Skill in Man and Machine«, Springer, New York, 1977
Mazollo, »Topics of Artificial Intelligence«, New York, Wien, Springer, 1976
Newell, Simon, »Human Problem Solving«, Englewood Cliffs, Prentice Hall, 1972
Nilsson, »Problem Solving Methods in Artificial Intelligence«, New York, McGraw Hill, 1971
Nilsson, »Principles of Artificial Intelligence«, Berlin, New York, Springer, 1982
Pearl, »Heuristics - Intelligent Search Strategies for Computer Problem Solving«, Addison Wesley, Massachusetts, 1984
Rich, »Artificial Intelligence«, McGraw Hill, New York, 1983
Siekmann, Wrightson, »The Automation of Reasoning«, Berlin, New York, Springer, 1983

Wilensky, »Planning and Understanding", Reading, AddisonWesley, 1983
Winograd, »Language as a Cognitive Process«, Reading, AddisonWesley, 1983
Winston, »Artificial Intelligence«, Reading, AddisonWesley, 1977
Winston, Horn, »LISP«, Reading, AddisonWesley, 1981
Wos, Overbeek, Lusk, Boyle, »Automated Reasoning: Introduction and Applications«, Prentice Hall, New Jersey, 1984

B. Deutschsprachige Fachliteratur

Bibel, Siekmann, »Künstliche Intelligenz«, Berlin, New York, Springer, 1982
Botvinnik, »Meine neuen Ideen zur Schachprogrammierung«, Berlin, New York, Springer, 1982
Bruderer, »Automatische Sprachübersetzung«, Wissenschaftliche Buchgesellschaft, 1982
Eisenberg, »Semantik und Künstliche Intelligenz«, Berlin, DeGruyter, 1976
Feigenbaum, »Die Fünfte Computer-Generation«, Birkhäuser, Stuttgart, 1984
Findler, »Künstliche Intelligenz und Heuristisches Programmieren«, Wien, New York, Springer, 1975
Graham, »Künstliche Intelligenz«, Luther Verlag, Sprendlingen, 1979
Hartwig, »Experimente zur Künstlichen Intelligenz mit C64/C128«, Markt & Technik-Verlag, München, 1987
Itzinger, »Methoden der Maschinellen Intelligenz«, München, Wien, Hanser, 1976
Kochen, »Kognitive Lernprozesse: Ein Erklärungsversuch«, New York, Springer, 1975
Savory, »Künstliche Intelligenz und Expertensysteme«, Oldenbourg Verlag, München, 1985
Schmitter, »Künstliche Intelligenz«, Hofacker, Holzkirchen, 1984
Slagle, »Einführung in die Heuristische Programmierung«, München, Moderne Industrie, 1972
Stachowiak, »Denken und Erkennen im kybernetischen Modell«, Wien, New York, Springer, 1969
Stede, »Einführung in die Künstliche Intelligenz Band 1«, Luther Verlag, Sprendlingen, 1983
Stede, »Einführung in die Künstliche Intelligenz Band 2«, Luther Verlag, Sprendlingen, 1984

Stichwortverzeichnis

Ein professionelles Grafikprogramm für Schneider CPC 6128 + Joyce

DR Graph

Präsentationsgrafiken mit professionellem Niveau

DR Graph ist ein interaktives Software-Paket, mit dem Sie Ihren Mikrocomputer zur Erstellung von Geschäftsgrafiken und Text-Charts verwenden können.

Die Fähigkeiten auf einen Blick:

- Linien-Grafiken, Histogramme, Torten-Grafiken, Stufen-Grafiken, Strich-Histogramme, Punkte-Grafiken und Text-Grafiken
- freie Wahl der Gestaltungselemente wie Beschriftungen, Titelzeilen, Legenden, Farben, Schriftarten und Ränder
- frei wählbare Skalierung
- variable Linien- und Balkenbreite





- Schnittstelle zu anderen Programmen
- beliebig positionierbare Anmerkungen
- Grafiken können gespeichert, geplottet oder gedruckt werden
- einfache Bedienung durch Menüauswahl

Hardware-Voraussetzungen: DR Graph läuft auf jedem Schneider CPC 6128 oder Joyce PCW 8256 mit einem oder zwei Diskettenlaufwerken. Die Grafiken können auf jedem Drucker oder Plotter ausgegeben werden, für den ein GSX-Treiber verfügbar ist. Dazu zählen Schneider-, Epson- und Shinwa-Drucker sowie der Plotter HP 7470A.

Bestell-Nr. 51614, 3-Zoll-Diskette

DM 199,–*

(sFr 178/öS 1990,-*)

* inkl. MwSt. Unverbindliche Preisempfehlung



Zeitschriften · Bücher
Software · Schulung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

705337

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Experimente zur künstlichen Intelligenz in BASIC auf CPC 464/664/6128

Der Autor:

OLAF HARTWIG studiert Informatik mit Schwerpunkt auf dem Gebiet der Künstlichen Intelligenz. Er wurde 1985 von Bundesforschungsminister Dr. Riesenhuber und Heinz Nixdorf für ein CAI-(Computer-Aided-Instruction-)Expertensystem als bester Programmierer mit der goldenen Diskette ausgezeichnet. Die gleiche Auszeichnung erhielt er erneut 1986 für ein neuartiges CAD-(Computer-Aided-Design-) Expertensystem.

Sind Maschinen intelligent? Können Computer denken? Erschließen Sie sich eines der interessantesten Gebiete der modernen Computerforschung! Anhand zahlreicher Programme erfahren Sie hier die Möglichkeiten und Grenzen der Künstlichen Intelligenz speziell auf dem Schneider CPC.

Das Buch gliedert sich in zwei Abschnitte: in einen Praxisteil und einen Abschnitt, der weiterführendes Hintergrundwissen zur Künstlichen Intelligenz vermittelt und ein umfassendes Forschungsprofil darstellt.

Der Schwerpunkt des Buches liegt auf der Praxis. Alle KI-Techniken werden durch anschauliche Programme vorgestellt. Sie können die vermittelten Informationen sofort auf Ihrem CPC nachvollziehen. Zusätzlich erhalten Sie jede Menge Anregungen zu eigenen Experimenten. Die KI-Programme können ohne weiteres in Ihre eigenen Programme integriert werden.

Aus dem Inhalt:

- Was ist Künstliche Intelligenz?
- Forschungsprofil der KI
- Verarbeiten natürlicher Sprache
- Verstehen geschriebener Sätze
- Der intelligente Dialog
- Parsing: Satzdekodierung
- Die Maschinenübersetzung
- Wissensrepräsentation
- KI und Robotics
- Der Micro-Maus-Wettbewerb
- Roboter der fünften Generation
- Das Bildverstehen
- Expertensysteme auf dem CPC
- Expertensystem-Folgerungssysteme
- Computer-Kreativität

Hard- und Software-Anforderungen:

Computersystem der CPC-Reihe

ISB N 3-89090-473-4





DM 49,-
sFr 45,10
öS 382,20

90473

Hartwig

Experimente zur künstlichen Intelligenz in BASIC auf CPC 464/664/6128

Markt&Technik

AMSTRAD CPC

OCR

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA



https://acpc.me/